



Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

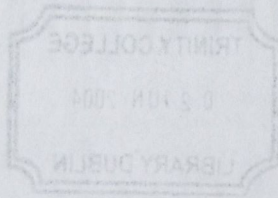
Time Bounded Medium Access Control for Ad Hoc Networks

Raymond Cunningham

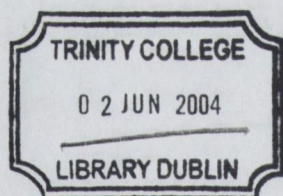
A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)



October 2003



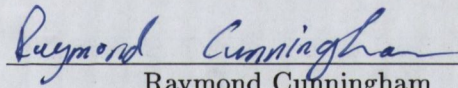
TH0815

7957

7496

Declaration

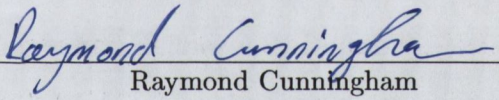
I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.


Raymond Cunningham

Dated: April 8, 2004

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.


Raymond Cunningham

Dated: April 8, 2004

Acknowledgements

Riding a roller-coaster [19] is jam packed full of highs and lows, twists and turns until you arrive at your destination, which is usually back where you originally started. Starting and trying to finish a PhD has a lot of similarities to riding a roller-coaster.

Having someone who is willing to accompany a starting PhD student on such an undulating and arduous journey is a terrific asset. Someone who keeps your feet on the ground during the highs, helps pick up the pieces during the lows and keeps a watchful eye for any approaching bends. I would like to thank my journey guide and supervisor Prof. Vinny Cahill for firstly allowing me to start on this journey in the Distributed Systems Group (DSG) in Trinity College Dublin and accompanying me along the way.

I would also like to thank my fellow roller-coaster riders in the DSG for being there for me at various stages of this journey. There are no words I could write that would do sufficient justice to the interesting/talented/funny group of individuals that composes the DSG. I have learned so much over the years, both technical and non-technical from members of the group. I don't think I could ever begin to repay their kindness, support and willingness to help.

I would like to say a special word of thanks to those individuals who were also just beginning, or had already started their own journey, as I started mine. You know who you are. Thank you for looking after me particularly as the roller-coaster was leaving the platform. A variety of roller-coaster technicians also deserve special praise for keeping my roller-coaster on track as the destination approached by checking drafts

and restoring backups. I would also like to thank those individuals whose paths crossed with mine while I was travelling on this journey and without whose contribution this thesis would never have been written.

Getting to the amusement park can be just as difficult as starting on any roller-coaster ride. Firstly, I would like to thank my three big brothers (even though one is younger than me) for their unquestioning support over the years. Secondly, I would like to thank my grandmother for the love and affection she has shown me throughout all the years of my life. Finally, I would like to dedicate this thesis to my parents. My father passed away before I had a chance to get to know him, everyone I meet who knew him only ever have positive and affectionate memories of him. My mother is the rock on which the Cunningham family is built, a constant source of unconditional love and support.

Raymond Cunningham

University of Dublin, Trinity College

October 2003

Abstract

The widespread adoption and deployment of wireless local area networking and in particular wireless ad hoc networking poses a number of challenging problems for distributed real-time applications use this wireless technology to communicate. These applications typically require guarantees on the timeliness of communication between the hosts running these applications.

An important building block in providing such timeliness to these hosts is to be able to give corresponding guarantees on the time required to access the wireless medium using the Medium Access Control (MAC) layer. Previous work on wireless multi-hop MAC protocols for ad hoc networks has focused on the twin goals of maximising throughput and minimising average packet delay as required for general-purpose applications.

In this thesis we describe a new MAC protocol for use in multi-hop ad hoc networks whose goal is to provide, with high probability, time-bounded access to the wireless medium for applications with guaranteed response time requirements. Secondary concerns of this new MAC protocol are maximising throughput and minimising average packet delay.

The Time-Bounded Medium Access Control (TBMAC) protocol is based on time-division multiple access with dynamic but predictable slot allocation. TBMAC uses a lightweight atomic multicast protocol to achieve distributed agreement on slot allocation and employs location information to minimise contention for slots. TBMAC is

the first time-bounded MAC protocol for multi-hop wireless ad hoc networks of which we are aware.

TBMAC provides a set of time bounds on gaining access to the wireless medium that depend on the current operational state of the protocol. The time bounds provided by TBMAC and their corresponding probabilities of being achieved have been derived mathematically and experimentally validated by means of a number of simulation experiments.

Contents

Acknowledgements	iv
Abstract	v
List of Tables	xiii
List of Figures	xiv
Chapter 1 Introduction	1
1.1 Ad Hoc Networking	2
1.2 Medium Access Control	5
1.3 Thesis	6
1.4 Issues not covered	7
1.5 Evaluation	8
1.6 Road Map	8
1.7 Summary	9
Chapter 2 Related work	10
2.1 Wired MAC Protocols	12
2.1.1 Background	13
2.1.2 Scheduled Access	16
2.1.3 Contention based Access	17

2.1.4	Analysis	18
2.2	Wireless MAC Protocols	19
2.2.1	Background	19
2.2.2	Contention-based Approaches	21
	Reducing Contention	22
	MACA	22
	MACAW	23
	802.11 Distributed Coordination Function	25
	Analysis	27
	Resolving Contention	28
	Black Burst	28
	Splitting Protocol	30
	Analysis	31
2.2.3	Schedule-based Approaches	31
	Centralised Scheduling	32
	Global System for Mobile communications	32
	802.11 Point Coordination Function	33
	Bluetooth	34
	Analysis	36
	Topology Dependent Scheduling	39
	Analysis	41
	Topology Transparent Scheduling	43
	Analysis	45
2.3	Agreement Protocols	46
2.3.1	Background	46
2.3.2	Rounds of Message Passing	48
2.3.3	Diffusion Protocols	50

2.4	Summary of Related Work	52
Chapter 3	A New MAC Protocol for Ad Hoc Networks	54
3.1	Protocol Basics	55
3.1.1	Cellular Structure	55
3.1.2	Periods of Operation	57
3.1.3	Time Bounds	58
3.2	Agreement Protocol	62
3.3	Protocol Details	65
3.3.1	Data Structures	65
3.3.2	Protocol Header	66
3.3.3	Non-empty CFP	69
	Requesting a slot	69
	Allocating a slot	70
	Deallocating a slot	71
	Leaving a cell	73
	Inter-Cell Communication	73
3.3.4	Empty CFP	74
	Bootstrapping Protocol Requirements	75
	Allocating a Slot	76
	Converging to a Consistent Allocation of Slots	77
	Convergence algorithm example	82
	Convergence algorithm analysis	84
	Deallocating a Collision Slot	85
3.3.5	Extensions	85
3.4	Protocol Inaccessibility	87
3.4.1	Modeling Arrival/Departures	88
3.4.2	Non empty cell	89

3.4.3	Empty cell	89
3.5	Summary	92
Chapter 4	Implementation	94
4.1	Implementation Environment	94
4.2	Initialisation	95
4.3	Slot management	97
4.3.1	Data structures	97
4.3.2	Querying the protocol	100
4.3.3	Allocating a slot	101
4.3.4	Deallocating a slot	106
4.3.5	Allocating/Deallocating an inter-cell slot	107
4.3.6	Atomic agreement	109
4.3.7	Centralised vs Distributed slot management	110
4.4	Communication	112
4.5	Summary	114
Chapter 5	Evaluation	115
5.1	Approach	116
5.2	Environment	117
5.3	Experimental Configuration	118
5.3.1	Movement pattern of mobile hosts	119
5.3.2	TBMAC parameters	121
5.3.3	Summary of parameters	122
5.4	Experimental Results	123
5.4.1	Expected Time Bound Probabilities	123
Non-empty cell		123
Empty cell		125

5.4.2 Obtained Time Bound Statistics 127

 Non-empty cell 127

 Empty cell 133

5.4.3 Other Results 134

 Inconsistent Allocation / Deallocation 134

 Failure to use allocated CFP slot 136

 Detection of unused CFP slot 137

5.4.4 Remarks 137

5.5 Summary 140

Chapter 6 Conclusion 141

6.1 Summary 141

6.2 Contribution 142

6.3 Perspectives 144

6.4 Future Work 144

Bibliography 146

List of Tables

2.1	Reviewed MAC protocol properties	53
5.1	Summary of values of experimental parameters investigated	122
5.2	Expected probability of a host choosing a distinct slot in the CP	125
5.3	Expected probability of a host choosing a distinct slot in the CFP . .	126
5.4	Comparison of obtained statistic and expected probability for a host choosing a distinct slot in the CP	130
5.5	Comparison of obtained statistic and expected probability of a host choosing a distinct slot in the CFP	134

List of Figures

2.1	Hidden Terminal Problem	20
2.2	Exposed Terminal Problem	21
3.1	Possible Cell and Channel allocation	56
3.2	Mobile host entering a cell in the CFP	60
3.3	CFP Slot Header	66
3.4	Adjoining Cells	74
3.5	Generated list of slots	77
3.6	Pseudo code for mobile host processing each 2-bit value in the slot bitmap of a bootstrap packet	79
3.7	Convergence of Slot Allocations	83
3.8	Subdivision of a Cell	86
4.1	Query/Change methods	95
4.2	Method to indicate a change in layout of neighbouring cells	96
4.3	Initialise/Retrieve the slot manager of the TBMAC protocol	97
4.4	Information related to a CFP slot	98
4.5	Retrieve a list of references to allocated slots	100
4.6	Retrieve a list of references to allocated inter-cell slots	101
4.7	Request the TBMAC protocol to allocate a slot	102
4.8	Retrieve a list of references to allocated slots	102

4.9	Requesting the allocation of a CFP slot	103
4.10	Delivering the allocation request for a CFP slot	105
4.11	Notify slot manager of the allocation of a slot	106
4.12	Monitor the allocation of a slot	106
4.13	Allocate a slot (assuming caller already has a slot allocated)	107
4.14	Deallocate a slot (assuming caller already has a slot allocated)	107
4.15	Requesting the deallocation of a CFP slot	108
4.16	Notify slot manager of the deallocation of a slot	108
4.17	Initialise/Retrieve the delay to delivery of the TBMAC protocol	110
4.18	Notify slot manager of delivery of atomic broadcast	111
4.19	Queue a packet for transmission by the TBMAC protocol	112
4.20	Pass packet to higher layer in protocol stack	113
4.21	Enable copy of packet to be passed to slot manager	113
4.22	Pass received packet to slot manager	113
5.1	Sample placement of mobile hosts at the start of an experiment	120
5.2	3 activated mobile hosts in an experiment	120
5.3	Expected probabilities of a host choosing a distinct CP slot	124
5.4	Expected probabilities of one or more hosts arriving within t seconds and each of these hosts choosing a distinct CP slot	124
5.5	Expected probabilities of a host choosing at least one distinct CFP slot	125
5.6	Expected probabilities of one or more hosts arriving within t seconds and each of these hosts choosing at least one distinct CFP slot	126
5.7	Obtained statistics for a host choosing a distinct CP slot	128
5.8	Difference between obtained statistics and expected probabilities for a host choosing a distinct CP slot	129
5.9	Obtained statistics for a host choosing at least one distinct CFP slot .	131

5.10 Difference between obtained statistics and expected probabilities for a
host choosing at least one distinct CFP slot 132

5.11 Percentage of inconsistently delivered slot allocation requests 135

5.12 Percentage of hosts that fail to use their slot 138

5.13 Percentage of failed slots detected as having failed 139

Chapter 1

Introduction

This thesis addresses the problem of providing time bounded medium access to applications with guaranteed response time requirements in multi-hop ad hoc networks through the design of a new medium access protocol called TBMAC. The TBMAC protocol uses time division multiple access (TDMA) with dynamic but predictable slot allocation. Dynamic allocation of TDMA slots is achieved through the use of an atomic multicast protocol. However, dynamic slot allocation requires the mobile hosts in a cell to have a consistent allocation of the TDMA slots allocated in the cell. Associated with the allocation of a TDMA slot are a set of time bounds that depend on whether the current allocation of CFP slots is consistent or not. In addition, the time bounds provided by this new protocol are not certain, therefore associated probabilities of achieving these time bounds are derived mathematically and then experimentally validated.

This introductory chapter presents relevant background information on ad hoc networks and medium access control in such networks that is necessary to understand the remainder of this thesis. The chapter also highlights the problem that this thesis is addressing, namely providing time bounded medium access to applications with guaranteed response time requirements. Issues that will not be covered in this thesis are

outlined in section 1.4. Towards the end of the chapter, a road-map for the remainder of this thesis is presented.

1.1 Ad Hoc Networking

Recent years have seen a rapid increase in research into and deployment of wireless networking technologies. Wireless networking can be divided into two distinct areas: infrastructure and ad-hoc (or infrastructure-less). In an infrastructure wireless network, there is a collection of fixed and mobile hosts. The fixed hosts are connected to each other by some form of communication network giving a static network topology or infrastructure. Communication from and to a mobile host in an infrastructure network is coordinated by one or more of these fixed hosts.

In contrast to wireless infrastructure networks, wireless ad hoc networking has focused on networks where all the hosts in the network are potentially mobile. In such ad hoc networks, the network topology is much more dynamic than in wired networks. In addition, depending on the wireless technology being used, there is an associated limitation on the transmission range.

This increased network dynamism and limited transmission range places a greater emphasis on mobile hosts to cooperate and coordinate with each other to allow communication to occur. For example, to allow a transmitting mobile host to communicate with another host outside its transmission range requires intermediate hosts to forward packets on behalf of the transmitting mobile host.

If all the hosts in the ad hoc network are within communication range of each, then the network is termed a single hop ad hoc network. In contrast to a single hop network, a multi-hop ad hoc network is one where one or more intermediate hosts forward the packets on behalf of a transmitting host to reach the packet's destination.

Some potential application domains for ad hoc networking include battlefield com-

munication, disaster relief (e.g., post-earthquake emergency services), mobile robotics, and inter-vehicle communication in intelligent transportation systems. Each of these application domains exhibit a number of different problems and requirements for ad hoc networking.

One problem is that the ad hoc network can become partitioned due to the mobility of hosts and the limit on the transmission range of the wireless technology being used. As an ad hoc network can partition, these partitions can then subsequently merge. This partitioning and merging would require cooperating mobile hosts to detect when the network is partitioning and merging to ensure that application state across the cooperating hosts remains consistent. For example, a platoon of soldiers becoming separated into different groups in a battlefield would need to maintain a consistent view of the platoon in the presence of such partitions.

A similar problem to detection of partitions, is that of how mobile hosts discover other mobile hosts in the network. An example of this would be when vehicles in an intelligent transportation system arrive at a junction and need to discover the presence or absence of other vehicles. To safely cross the junction, all the vehicles at the junction require a consistent view of all the other vehicles at the junction.

In each of the above application domains, there is an obvious requirement for coordination between mobile hosts. For example, mobile robots coordinating with each other to assign application specific tasks that need to be carried out. Coordinating rescue workers after an earthquake has occurred, to ensure critical tasks are carried out, is another example of this coordination requirement.

However, providing this coordination is difficult due to the lack of a central point to coordinate the actions of the mobile hosts in the network. Reliance on such a centralised coordinator is unattractive if one or more mobile hosts are in a partition that does not contain this coordinator. Therefore, coordination/cooperation between a set of mobile hosts in the ad hoc network needs to be carried out in a distributed

peer-to-peer manner. If this coordination were to rely on a particular mobile host, then other mobile hosts would then need to monitor whether that particular host is still within communication range of the set of hosts.

A requirement closely related to coordination/cooperation between a set of hosts is the need to ensure the consistency of replicated data across a collection of mobile hosts. In other words, to allow mobile hosts to coordinate their actions correctly requires that the information these mobile hosts use to make their decision be consistent across the mobile hosts.

The requirements in the above paragraphs for coordination/cooperation between mobile hosts and consistent maintenance of replicated data across a collection of mobile hosts are examples of functional requirements. An important non-functional requirement is the Quality of Service (QoS) or timeliness guarantees in these application domains.

An example of this timeliness requirement in the intelligent transportation systems domain is when the lead vehicle of a platoon of vehicles applies its brakes to avoid an obstacle. The lead vehicle would be required to inform the other vehicles within a known time bound to allow the following vehicles to also avoid the obstacle. In a battlefield scenario, timely communication between the soldiers of a platoon in combat could allow soldiers to locate and recover wounded colleagues. In an advanced factory automation system, mobile robots would need to communicate in a timely fashion to prevent accidents from occurring.

To meet this timeliness requirement, mobile hosts need to coordinate and cooperate to ensure that their transmissions on the shared wireless medium are successful. Timely communication needs to be maintained even in the presence of network partitions and mobile hosts joining and leaving the ad hoc network. Each layer in the protocol stack needs to provide timeliness guarantees on the transmission and reception of packets at that layer. In particular, the medium access control (MAC) layer should provide

timeliness guarantees to layers above it. The focus of this thesis is the design of a new MAC protocol meeting these timeliness requirements for the application domains highlighted above.

1.2 Medium Access Control

A medium access control (MAC) protocol consists of a well defined set of actions that a set of hosts agree upon to coordinate transmissions on some shared communication medium. Communication mediums can be either wired (all hosts are connected to a shared “wire” e.g., twisted pair copper cables) or wireless (e.g., infra-red or radio frequency).

The set of actions executed by the hosts using a MAC protocol are designed to meet the goals of that particular protocol. One important goal of many MAC protocols is to reduce or eliminate the possibility of collisions occurring due to simultaneous transmission of data/packets by two or more hosts in the network. Other typical goals of MAC protocols include maximising the overall data throughput of hosts in the network, minimising the average packet delay or providing guarantees on the delay before each host can transmit on the medium.

As described in the previous section, the topology of an ad hoc network can be constantly changing. This dynamic network topology causes a number of associated problems for MAC protocols designed for ad hoc networks.

Firstly, coordinating access to the wireless medium for a set of mobile hosts within communication range of each other is difficult to achieve if the set of mobile hosts is constantly changing due to mobility. This problem is exacerbated when the ad hoc network can partition and merge. Secondly, a problem related to coordinating access to the wireless medium is how the state information of the MAC protocol, replicated at each mobile host, is maintained. Thirdly, as communication is done over a wireless

medium, there is a possibility of interference [64] due to obstacles in the environment and also due to mobile hosts just outside the transmission range of a mobile host in a multi-hop ad hoc network (the well-known hidden terminal problem [81]). This increased possibility of interference will result in a corresponding increase in the number of packets corrupted during transmission.

Most research (e.g. [42], [35], [9]) into medium access in multi-hop ad hoc networks has focused on the twin goals of maximising the overall throughput of mobile hosts in the network and minimising the average packet delay for each mobile host. The focus in these research projects is not to provide each host with access to the medium in a time bounded manner. In such MAC protocols, discussed in Chapter 2, there is a possibility that a mobile host's time to access the medium is unbounded. Other researchers (e.g [74], [52]) have focused on providing such a time bound in single hop ad hoc networks. As discussed in the previous section, providing such a time bound in multi-hop ad hoc networks is an important building block for a number of application domains.

1.3 Thesis

This thesis presents a new MAC protocol, called the Time Bounded Medium Access Control (TBMAC) protocol, for use in multi-hop ad hoc networks. Applications where the use of TBMAC is appropriate are those applications with guaranteed response time requirements.

As highlighted in sections 1.1 and 1.2, providing such guarantees to each host in an ad hoc network is a difficult problem. To address this problem, the TBMAC protocol uses time-division multiple access (TDMA) with dynamic but predictable slot allocation. TBMAC uses a lightweight atomic multicast protocol to achieve distributed agreement on the allocation of slots to mobile hosts and employs location information

to minimise contention for slots.

However due to the dynamic topology of an ad-hoc network, it is difficult to provide time bounds with certainty on the allocation/deallocation of TDMA slots. Therefore, the TBMAC protocol provides probabilities for a given time bound being obtained. Parameters to the TBMAC protocol can then be chosen to ensure that the probabilities of these time bounds being obtained are sufficiently high.

By providing an atomic multicast protocol, TBMAC addresses the problems of coordinating between mobile hosts highlighted in the previous section and maintaining consistency replicated information, e.g. slot allocation, across a number of mobile hosts. TBMAC maintains this replicated information consistently in the presence of the network partitioning/merging and of mobile hosts joining and leaving the ad hoc network. TBMAC uses location information to reduce the problem of interference from neighbouring mobile hosts.

1.4 Issues not covered

Providing time bounds to different classes of applications using ad hoc networking is a difficult task that impacts on every layer of the protocol stack from the MAC layer up to and including the application layer. The goal of the TBMAC protocol is to provide bounds on the time to access the wireless medium at the MAC layer.

These time bounds are one building block in providing end-to-end quality of service to applications in ad hoc networks with guaranteed response time requirements. To build upon these time bounds at the MAC layer, higher layers above TBMAC would need to make appropriate design decisions to provide end-to-end quality of service to applications.

For example, a routing layer would need to ensure that a certain quality of service is enforced on the discovery and maintenance of routes between hosts in the ad hoc

network. Other functionality closely related to quality of service is admission control. TBMAC does not make quality of service or admission control decisions on behalf of the application.

TBMAC does provide functionality to allow higher layers to make such quality of service or admission control decisions. For example, higher layers can query the TBMAC protocol for the number of TDMA slots currently allocated and then allocate/deallocate a number of slots to meet higher layer requirements. The design of layers above TBMAC will not be discussed further during the remaining chapters of this thesis. However, functionality provided by the TBMAC protocol to enable higher layers to customise the operation of TBMAC to their specific requirements will be highlighted.

1.5 Evaluation

As the TBMAC protocol provides a set of time bounds on gaining access to the wireless medium that depend on the operational state of the protocol. The evaluation of the TBMAC protocol are designed to experimentally verify the probabilities for each of these time bounds that will be derived in chapter 3.

1.6 Road Map

The remainder of this thesis is structured as follows

Chapter 2 presents the state of the art in medium access control for ad hoc networks.

This chapter also covers relevant wired medium access control protocols along with a brief overview of protocols to allow agreement to be reached between a set of hosts.

Chapter 3 presents the design of the TBMAC protocol and derives a set of time bounds and their associated probabilities for this protocol.

Chapter 4 presents an instantiation of the design of the TBMAC protocol from the previous chapter.

Chapter 5 experimentally validates the time bounds derived in Chapter 3.

Chapter 6 summarises and discusses future work.

1.7 Summary

This chapter outlined the goals of this thesis and also highlighted issues that will not be addressed by this work. Background information on ad hoc networks and medium access control, relevant to the rest of the thesis, was also presented.

Chapter 2

Related work

As this thesis is concerned with the design of a new time bounded medium access control (MAC) protocol for multi-hop ad hoc networks, this chapter reviews some important wired and wireless MAC protocols that have been developed over recent years. The important features of each MAC protocol will be described, followed by an analysis of how each protocol addresses three important networking characteristics:

1. Timeliness
2. Agreement
3. Network Topology

The timeliness property of a MAC protocol refers to the timeliness guarantees on the delay before a mobile host gains access to the wireless medium and is then able to transmit its message/packet. As we shall see over the following sections, there are some MAC protocols (both wired and wireless) whose goals do not include providing such a timeliness guarantee. Such MAC protocols aim to provide fairness in the way that hosts utilise the medium. In contrast to these MAC protocols, other MAC protocols have been specifically designed to provide a timeliness guarantee as one of their design goals.

A property closely associated with the timeliness property of a MAC protocol is whether or not the MAC protocol depends on or provides some form of agreement between hosts participating in the protocol on important MAC protocol related information. This agreement is typically not needed in MAC protocols that do not provide a timeliness guarantee. Obviously, agreement protocols can be built on top of any MAC protocol, however, ensuring that agreement is reached in a time bounded manner is difficult if the underlying MAC protocol does not provide a guarantee on the time taken to transmit a message. In MAC protocols that do provide a timeliness guarantee, this agreement can be reached before the protocol begins execution. For example, in TDMA (see section 2.2.1), the slot (or slots) that each host uses to transmit its messages is agreed offline before the protocol starts executing.

Traditionally in wired networks, network topology is relatively static. A MAC protocol determines how a set of hosts gain access to a shared medium (i.e., the cable connecting the hosts). Larger scale networks would then be built up by joining collections of these single hop networks. The network topology thus consisted of a collection of single hop networks. Repeaters or gateway hosts connecting two or more single hop networks are responsible for relaying messages on behalf of hosts connected to one network.

Wireless networks have a much more dynamic network topology due to the mobility of some or all of the hosts in the network. Ad hoc networks could be seen as the most extreme example of dynamism in such wireless networks with the possible mobility of all hosts in the network. In ad hoc networks, a host is limited by the transmission range of its wireless interface. If a host wishes to communicate beyond its transmission range, then one or more of the hosts within its communication range would need to act as a router on behalf of this node. This contrasts with wired networks as specific devices are given the task of acting as a repeater/gateway/router.

Section 2.1 gives a brief background on the historical development of wired MAC protocols followed by a review of some important wired MAC protocols. Following this, section 2.2 presents a similar review of wireless MAC protocols after providing a primer on the differences between wired and wireless communication. In section 2.3, background information on distributed agreement is covered followed by a description of a number of related protocols. This chapter concludes with a summary in section 2.4.

2.1 Wired MAC Protocols

Wired MAC protocols can be divided into two categories: broadcast based and point-to-point. As point-to-point MAC protocols are typically used over large distances (wide area networks), these MAC protocols will not be covered in this chapter as this thesis is primarily concerned with timely medium access in ad hoc networks which are predominantly broadcast based.

Over the next three sections, important background/historical information related to medium access in wired networks is covered followed by reviews of two influential real-time MAC protocols for wired networks developed in recent years. Typically, there are two main approaches to medium access in a wired network:

1. Scheduled access
2. Contention based access

Hosts using scheduled access negotiate amongst themselves when each should transmit so that transmissions from these hosts avoid collisions as much as possible. In contrast, in contention-based access hosts with packets to transmit compete with other such hosts to gain access to the medium. This contention occurs before the transmission of each packet. The two real-time MAC protocols covered later in this section provide an example of each of these approaches.

2.1.1 Background

As mentioned in the previous section, medium access can be either scheduled or based on contention. Scheduled access can be further broken down into Frequency Division Multiple Access (FDMA) or Time Division Multiple Access (TDMA) [75].

In FDMA, each host connected to the medium is assigned a frequency to use. If a host wishes to receive a transmission from another host then it switches to the frequency allocated to the transmitting host. One example of FDMA is the transmission of television over the UHF and VHF frequency bands.

In TDMA, access to the medium is divided into discrete time intervals, called slots. Each host is then statically allocated a slot (or slots) in which to transmit. This allocation of slots repeats after the time of the last slot has passed. When a host wishes to transmit, it waits until its slot arrives and then transmits. To divide access to the medium into time slots requires the clocks of the hosts to be synchronised to within a known precision of each other.

The main advantage of both FDMA and TDMA is that hosts have deterministic and time-bounded access to the wired medium since in FDMA a host has its own frequency and in TDMA a host is allocated its own slot that repeats periodically. The main disadvantage of both FDMA and TDMA is that there is no mechanism in place to allow a host that wishes to transmit more data than the capacity allocated to it (frequency in FDMA or slot in TDMA). Another disadvantage of each of these protocols is that hosts need to agree on the allocation of frequencies or slots before the protocol starts, thus placing an upper bound on the number of hosts that can participate in the protocol.

In contrast to these schedule-based access approaches, in contention based access MAC protocols hosts compete with each other to transmit. One of the first contention based schemes was the ALOHA protocol developed by researchers at the University of Hawaii in the 1970s [5]. Although the ALOHA protocol was originally developed for

radio broadcasting, the medium access approach used has influenced subsequent wired MAC protocols. The ALOHA protocol is very simple with hosts transmitting whenever they have data to send. If two packets collide, then each sender waits a random amount of time before transmitting again. The main disadvantage of the ALOHA protocol is that the maximum possible utilisation of the medium is only 18% [76] of the total capacity for an arbitrary number of hosts with transmissions of fixed length packets following a Poisson distribution. The underlying reason for this low utilisation is that the a host can transmit at any time, therefore any of its packets can be corrupted at any point during transmission.

To address this problem, slotted ALOHA was introduced which has twice the capacity (36%) of the ALOHA protocol [66]. In this protocol, access to the medium is divided into slots in a similar way to TDMA. Instead of statically allocating slots to hosts, when a host has data to send, the host waits for the next slot to begin and transmits in this slot.

To further increase the number of successful transmissions, a number of carrier sense protocols have been developed [80]. Carrier sensing is possible when the host can detect if the medium is busy or not. In Carrier Sense Multiple Access (CSMA) protocols, a host with data to transmit first senses the medium to check if there is any transmission ongoing. When a host detects that the medium is idle, the host then transmits its data. If the medium is busy, the host has a number of options. In non-persistent CSMA, the host waits a random amount of time before checking whether the medium is idle or not. If it is, the host transmits. In 1-persistent CSMA, the host senses the medium until it becomes idle and then transmits. A variation of 1-persistent CSMA is p-persistent CSMA which is used with slots. When a host has data to send and the medium is idle in a slot, the host transmits with probability p or defers until the next slot with probability $1 - p$. Each of these CSMA approaches improve upon slotted ALOHA with the performance of the p-persistent CSMA varying with the p

parameter [79].

An improvement over these CSMA protocols is a 1-persistent CSMA MAC protocol that allows hosts to detect collisions when they occur and then to cease transmission. This MAC protocol is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). Hosts detect collisions by monitoring the received signal from the cable and comparing it to the transmitted signal. On detection of a collision, a host transmits a jamming signal to inform all hosts of the collision. The host then waits a random amount of time before attempting to transmit again.

The most well-known CSMA/CD protocol is the IEEE 802.3 (Ethernet) standard. The random amount of time chosen in the Ethernet protocol when a collision occurs is calculated using the binary exponential back-off (BEB) algorithm. This algorithm uses the worst case time, called the slot time, to detect a collision. After the first collision, each host generates a number either 0 or 1 and waits this number of slot times. After waiting, if the medium is still idle then the host attempts to transmit again. If another collision occurs, a number, n , is generated in the range $0 \leq n \leq 2^i - 1$ where $i = 2$ and the host waits n slot times. If collisions continue to occur, then the value of i is multiplied by 2 after each collision up to a maximum of 10 collisions. By detecting collisions, CSMA/CD also improves upon slotted ALOHA [79] [71].

None of the above contention based MAC protocols provide hosts with time-bounded access to the wired medium. These protocols were designed to improve the total network throughput of all the participating hosts. As none of the protocols provide time bounded access, there is little need for agreement between the hosts. Slotted ALOHA is the only MAC protocol that requires the clocks of individual hosts to be synchronised so that hosts agree on the starting time for each slot.

2.1.2 Scheduled Access

An example of a hard real-time system using scheduled access is the Time-Triggered Protocols (TTP) developed at the Institut Für Technische Informatik at the Technische Universität Wien [47], [46]. TTP participants use TDMA with synchronised clocks [48] to access a shared (possibly replicated) medium. TTP builds on top of TDMA to provide a fault tolerant membership service to higher layers. .

Two versions of TTP have been developed. TTP/C is designed for the implementation of fault tolerant distributed systems while TTP/A is designed for non fault tolerant field bus applications. In the following paragraphs, the version described is the TTP/C fault tolerant version.

As TTP uses TDMA, every TTP node is statically assigned a time slot on each medium. A TTP node consists of two parts, the communication controller and the host. A collection of TTP nodes connected to a shared replicated medium is called a cluster. The communication controller is independent of the software running on the host. Information is transferred between the communication controller and the host using the communication network interface (CNI). The communication controller maintains a static Message Descriptor List (MEDL) that indicates when a message needs to be sent or received from the underlying medium.

Each TTP node maintains a membership vector which represents the maximum number of TTP nodes in the cluster. A TTP node is marked as either functioning correctly or not in this membership vector. Updates to this membership vector are done using a reliable fault tolerant multicast protocol [33].

For example, to detect the failure of a TTP node, each TTP node transmits in the slot allocated to it on each replicated media. At the end of a TTP round, each correctly functioning TTP node creates a bitmap marking those slots that it received correctly and incorrectly in the previous round. The TTP node then includes this bitmap in each of its transmissions in the following round. At the end of the following

round, if a majority of TTP nodes deem that a particular TTP node has failed based on the recently received bitmaps then each correctly functioning TTP node updates their membership vector accordingly. The TTP node deemed to have failed refrains from transmitting in its allocated slot until it is repaired.

2.1.3 Contention based Access

An example of a real-time contention based MAC protocol is the Controller Area Network (CAN) [65] which has gained widespread acceptance in factory automation and the automotive industry.

CAN is a bit oriented MAC protocol where each host transmits a series of recessive and dominant signals over a twisted pair cable. The CAN medium being idle is equivalent to each host transmitting a recessive signal. A host transmitting a dominant signal when all other hosts are transmitting recessive signals causes the “recessive” hosts to defer. For simplicity, a recessive signal can be thought of as a 1 and a dominant signal can be thought of as a 0.

A message, called a data frame in CAN, can vary in length up to a maximum length and it consists of a number of fields. Firstly, there is the Start of Frame field which consists of a dominant signal. The Start of Frame field is only transmitted by a host when the medium is idle. Following the Start of Frame is the Arbitration field which is used to determine which host gains access to the medium. After the Arbitration field are the Control, Data, CRC and ACK fields, followed by the End of Frame field.

The Arbitration field is broken into two parts. The first part, called the Identifier, represents the priority of the message being transmitted. The second part, called the Remote Transmission Request Bit, is used to differentiate between Data frames and a request by a host for another host to transmit data to it.

If two or more hosts transmit at the same time, each host monitors each bit in the Identifier subfield as it is transmitted. If a host transmits a recessive signal but detects

that there is a dominant signal on the wire, then this transmitting host defers. Thus the frame with the lowest value in its Identifier subfield (i.e. the highest priority) gains access to the medium to transmit the remaining fields. Note that if two hosts transmit with the same priority then a garbled frame will result.

2.1.4 Analysis

As the TTP protocols use TDMA as their medium access strategy, TTP suffers from similar problems to TDMA where the bandwidth allocated to a TTP node is fixed when the cluster is created. To address some of the goals of designers, TTP uses a reliable multicast protocol to overcome faults during transmission on the medium or faults due to failed TTP nodes. To construct larger scale networks, TTP uses gateways to connect two or more clusters.

In CAN, hosts with the highest priority have time bounded access to the CAN network. However, if two hosts transmit with the highest priority at the same time then their transmissions will collide. To prevent this from occurring, highest priority CAN messages need to be scheduled so that they do not collide.

One approach by Livani and Kaiser [49] is to reserve a time slot for each highest priority message. Information about the time slots is maintained in a calendar that is replicated at each host connected to the CAN network. Updates to this replicated calendar must be done in a consistent way across each host. For example using an atomic broadcast protocol [40]. A number of researchers ([58] and [60]) had assumed that successful message transmission in CAN was atomic and totally ordered. However, this assumption about the atomicity and ordering of CAN messages has been shown not to hold [69].

2.2 Wireless MAC Protocols

In a similar way to wired MAC protocols, contention based and scheduled access are the two main approaches used to gain access to a wireless medium. The wireless MAC protocols covered in this section will be divided based on which of these approaches they use. During the course of this section, whether a MAC protocol is designed for infrastructure or ad hoc networks will be highlighted. In the next section, some of the problems related to wireless communication are described.

2.2.1 Background

Unlike wired communication where the network topology is typically stationary and thus communication is more easily predictable, in contrast wireless network topology is much more dynamic due to mobility of hosts in the network making communication more difficult [64]. Depending on the wireless technology being used, there is an associated limitation on the transmission range of a mobile host. The quality of signal from a transmitting host degrades as the signal travels, thus bounding the distance over which a mobile host can transmit. Another problem is that this bound on the distance is variable, depending on interference effects from the environment such as multipath fading [64].

As a consequence of the limited transmission range of a mobile host, a number of communication problems arise. The hidden terminal problem [81] is an example of one such problem that can arise. This problem can occur in both infrastructure and ad hoc wireless networks and is illustrated in figure 2.1. In this figure, host A is currently transmitting to host B. As host C is outside of A's communication range, it does not detect the transmission of A and also begins transmitting to host B. This results in the transmission of both A and C colliding at B. In this case, host C would be considered a hidden terminal from mobile host A's perspective.

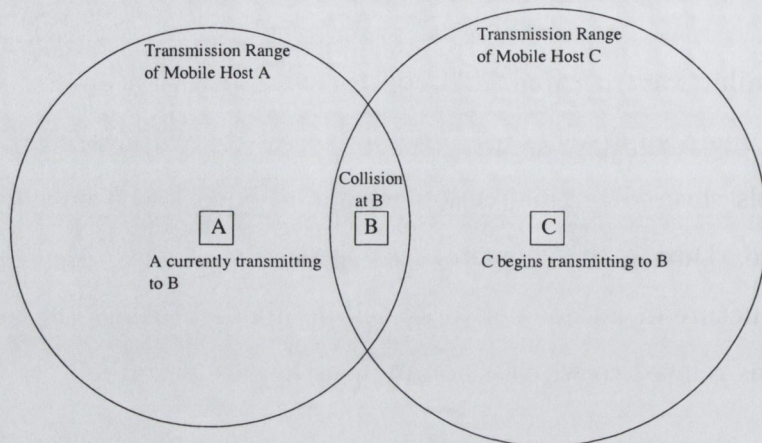


Fig. 2.1: Hidden Terminal Problem

As can be seen from figure 2.1, collisions occur at the receiver. Therefore, the two transmitting hosts, A and C, are unable to detect that their transmissions are causing a collision.

A collision could also happen when A and C are within communication range of each other if they both begin transmitting at exactly the same time. The underlying reason for the inability of A and C to detect such collisions is because mobile hosts are typically only equipped with one antenna. The antenna can be used either to transmit or to receive transmissions (the antenna cannot do both simultaneously). Therefore, typical wireless communication is half-duplex. Carrier sensing is possible only before transmission has begun.

It is possible to build full-duplex radios. However, a problem with such radios is that they are typically more expensive to build. Another problem relates to the capture effect specific to wireless communication [64]. As stated previously, the transmitted signal attenuates with distance. Mobile hosts that are sufficiently close to the transmitter can be “captured” by that transmitter’s radio. In other words, capture occurs when a receiver is within communication range of two transmitting hosts but is able

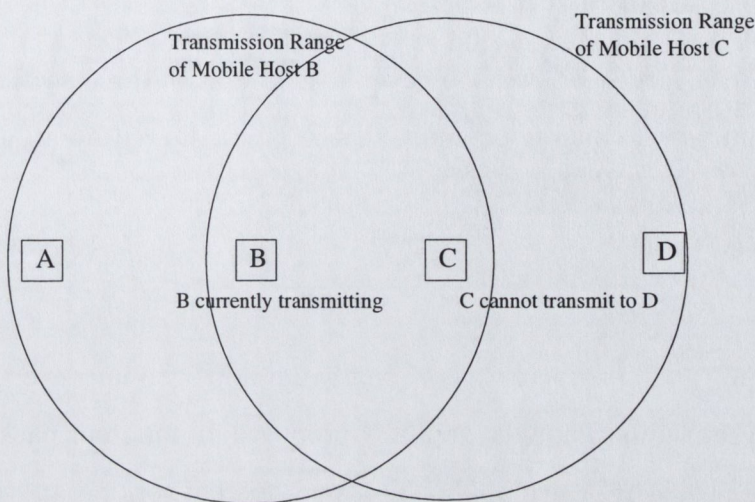


Fig. 2.2: Exposed Terminal Problem

to cleanly receive only the signal from the closer transmitter. This is equally true for a full-duplex radio which could potentially capture itself!

A related problem to the hidden terminal problem, called the exposed terminal problem, can occur in ad hoc networks (see figure 2.2). Here, a potential transmitter, C, defers after sensing the transmission of another mobile host, B, even though the transmission by C would not cause the reception of C's packet at D to be corrupted. In this case, C would be the exposed terminal.

2.2.2 Contention-based Approaches

In each of these two approaches, mobile hosts contend with each other for access to the wireless medium when they have data to transmit. The first set of MAC protocols attempt to reduce the possibility of contention occurring and also to reduce the possibility of contention recurring when it does occur. The second set of MAC protocols try to resolve contention when it occurs and allow one mobile host gain access to the wireless medium.

Reducing Contention

Three MAC protocols will be covered in the next three sections whose goal is to reduce contention when it occurs. This will be followed by an analysis of each of these protocols.

MACA

The MACA [42] protocol was originally developed to improve the performance of amateur packet radio. Though originally proposed for amateur packet radio, the techniques proposed by MACA are applicable to ad hoc networking and have been subsequently incorporated into a number of MAC protocols for ad hoc networks.

In MACA, carrier sensing of the wireless medium before transmitting is not done. Each mobile host transmits a "Request To Send" (RTS) control packet causing mobile hosts that receive this packet to defer. The destination of the RTS packet replies with a "Clear To Send" (CTS) control packet. In a similar way to the RTS packet, the CTS packet inhibits other hosts from transmitting allowing the originator of the RTS packet to transmit its data to the destination. If a transmitting mobile host does not receive the expected CTS packet correctly then a binary exponential back-off algorithm is executed.

To allow this RTS/CTS packet exchange to occur, every mobile host must wait an agreed amount of time, the slot time, after both the RTS and CTS packet. The transmitting host also includes in its RTS packet the amount of data that it wishes to transmit. The destination also includes this amount in the CTS when the destination replies. Listening mobile hosts will then be able to determine how long to defer their transmission.

Using the above scheme, there is still a possibility of either the RTS or CTS packets being corrupted due to collisions (i.e., the hidden terminal problem). However, since the size of RTS and CTS packets is relatively small compared to a data packet, mobile

hosts executing the RTS/CTS packet exchange reduce the possibility of the larger data packets being corrupted due to a hidden terminal. This reduces but does not eliminate the possibility of the larger data packets being corrupted. If a hidden terminal does not correctly receive the CTS packet, then a transmission by the hidden terminal after the CTS packet could corrupt the transmission of the larger data packet.

MACA only partially addresses the exposed terminal problem since hosts in MACA do not sense the medium before transmitting. For example, if an exposed terminal does not hear the CTS reply to a previous RTS packet, then the exposed terminal can begin transmitting. If, however, the exposed terminal does receive the CTS packet correctly, then it defers its transmission.

MACAW

The MACAW [10] protocol builds on MACA and was developed for use in a wireless infrastructure network at the Xerox Corporation's Palo Alto Research Center. The infrastructure network developed consists of base stations, installed in the ceiling, and custom built portable devices, called pads. All wireless communication between the pad and the base station was over a single 256kbps channel. The base stations are connected to each other by an Ethernet.

Both the base stations and the pads transmit at the same signal strength and the transmission range is approximately 3 to 4 metres. Such a small transmission range causes the cell around a base station to be approximately 6 meters in diameter. The signal strength decays rapidly beyond this transmission range bound, this reduces both the possibility of interference (other than collisions) from other transmitters and of capture.

MACAW in a similar way to MACA ignores sensing of the wireless medium before transmitting and it also uses the RTS/CTS handshake to improve the possibility of correct transmission of data. MACAW extends MACA in two ways. Firstly, MACAW

introduces two extra packets to assure correct reception of messages and to reduce the possibility of collisions. MACAW introduces an acknowledgement packet after the transmission of the data packet to assure the transmitter of the data packet that it was received correctly. The underlying reason behind the introduction of this acknowledgement packet is due to the greater possibility of a wireless transmission being corrupted due to transient errors compared to wired transmissions.

The second extra packet that MACAW introduces is the Data-Sending (DS) packet that is sent immediately after the RTS/CTS exchange to indicate to mobile hosts that this exchange was successful. When a listening mobile host receives the DS packet, then it defers until after the following data packet has been transmitted. The DS packet is introduced to prevent a host that does not correctly receive the CTS packet from beginning to transmit and thus potentially corrupt the transmission of the data packet.

If carrier sensing was used in MACAW, then a DS packet would be unnecessary as a mobile host would first sense if the transmission of the data packet was ongoing and thus would defer its transmission. The DS packet does not address the hidden terminal problem, as it is a packet originated at the transmitter and therefore does not prevent collisions at the receiver. The DS packet does not address the exposed terminal problem since an exposed terminal will defer its transmission until after the transmission of the data packet.

The second way that MACAW improves MACA is by changing the backoff algorithm used after the transmission of a packet. The backoff algorithm, binary exponential backoff (BEB), used in MACA is exactly the same backoff algorithm used in Ethernet. A problem with the BEB algorithm, highlighted by the designers of MACAW, is that one host can begin to dominate while the other hosts are backed off when the network is heavily loaded. A host that successfully transmits a packet, resets its backoff counter to a minimum value while a host that is unsuccessful continues to

increase its backoff counter up to a maximum. This implies that the successful host has a much greater chance of continuing to be successful when it wishes to transmit another packet compared to a host that increases its backoff counter.

MACAW attempts to alleviate this problem by changing the backoff algorithm to include the current value of the backoff counter in the header of the transmitted packet. When a host correctly receives a transmission, it sets its local backoff counter to the value in the transmitted packet. If all hosts are within communication range of each other, the copying of the backoff counter ensures that all hosts are using the same backoff counter leading to a fairer allocation of bandwidth amongst the hosts.

802.11 Distributed Coordination Function

Recent years has seen widespread adoption and use of the wireless LAN standard from the IEEE [35]. This rapid growth and deployment is expected to continue [4] over the coming years. The 802.11 standard specifies both the physical layer (infra red, frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS)) and the MAC layer.

Both the FHSS and the DSSS part of 802.11 operate in the 2.4GHz frequency band. In FHSS, the 2.4GHz frequency band is divided into 79 channels, each 1MHz wide. FHSS transceivers hop pseudo-randomly between these 79 channels. In DSSS, there are 13 different spreading codes defined. In the 802.11 standard, two spreading codes, whose center frequencies are at least 30MHz apart, can operate simultaneously without interference. At the current time, the author is unaware of any products implementating the infra red physical layer of the 802.11 standard.

The 802.11 MAC layer is designed for medium access in both infrastructure and ad hoc networks. The Point Coordination Function (PCF) is the part of the 802.11 MAC layer that deals with medium access in infrastructure networks and will be covered in section 2.2.3. In contrast, the Distributed Coordination Function (DCF) deals with

medium access in ad hoc networks.

Like MACA and MACAW, the DCF uses an RTS/CTS packet exchange. The DCF allows a size threshold to be specified, above which data packets will use the RTS/CTS handshake and below which the data packet will be simply transmitted. When a mobile host receives an RTS addressed to it, the mobile host responds with a CTS packet within a known period of time called the Short Inter-Frame Space (SIFS). An acknowledgement to the successful transmission of a data packet is also sent within this SIFS time period. The size of the SIFS time period is related to the underlying propagation delay of the physical layer being used.

Unlike MACA and MACAW, the DCF of 802.11 uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to determine whether or not a mobile host gains access to the wireless medium. As noted in section 2.2.1, collision detection is impossible to achieve with the half-duplex radios assumed by 802.11. Instead, the DCF utilises collision avoidance whereby a host waits for a random amount of time before attempting to transmit. Each host uses a contention window (CW) and generates a random number between zero and CW as a counter value. The host then decrements this counter value by 1 after waiting a DCF Inter-Frame Space (DIFS) period of time. If while waiting, the host senses another transmission beginning, then the host stops its countdown until this transmission has completed. When the counter value reaches zero, the host begins the transmission of its packet. The DCF doubles the size of the CW (up to a maximum value) when it fails to receive a CTS packet after an RTS packet or when it fails to receive an acknowledgement for a data packet transmitted without the RTS/CTS handshake. The SIFS is a shorter time period than the DIFS, this allows a CTS packet or an acknowledgement to a data packet to preempt the transmission of another RTS or data packet.

Analysis

The goal of each of the MAC protocols described in the previous section is to reduce the possibility of collision and corruption of data and thus increase the total network throughput. None of these protocols tries to address the issue of providing any guarantees on the maximum time before a mobile host can transmit a message.

A host that has a number of packets to transmit must continually contend with other hosts in the network. In both MACA and MACAW, RTS control packets can easily be corrupted due to two or more mobile hosts transmitting at approximately the same time. The possibility of this occurring in 802.11 is reduced (but not eliminated) by each mobile host sensing the medium for a random number of time periods before transmitting. The RTS/CTS packet exchange is designed to increase throughput by reducing the possibility of corruption of larger data packets.

In these schemes, mobile hosts do not try to reach agreement to coordinate transmission of their packets. Reaching such an agreement would avoid packet collisions but would reduce network throughput since agreement would require an exchange of packets between the mobile hosts. Avoiding such an agreement prior to transmission of packets greatly simplifies the design of the above protocols.

In terms of network topology, the above protocols are suited to medium access in multi-hop ad hoc networks. The DCF of 802.11 was specifically designed with such networks in mind. Both MACA and MACAW were not originally designed with multi-hop ad hoc networks in mind. However the underlying principles of these two protocols have been influential in more recent ad hoc MAC protocols such as the FAMA family of MAC protocols [29] and IEEE 802.15 (Zigbee) [6]. These later protocols will not be presented as they have similar properties to MACA, MACAW and the DCF of 802.11.

A number of researchers [55], [50], [57] have built on top of the DCF of 802.11 in an attempt to ensure fairness between the hosts in the network. These approaches change the way that packets are queued in an attempt to increase the average network

throughput of each host. Other researchers [41] have attempted to provide priority scheduling of packets on top of the DCF by changing the RTS/CTS handshake to include the priority of the next packet to be transmitted. In each of these approaches, no time bound on accessing the wireless medium is given.

Resolving Contention

An alternative approach to reducing contention is for the MAC protocol to resolve this contention between competing hosts when it arises. Two MAC protocols will be covered in the next two sections whose goal is to resolve contention when it occurs. This will be followed by an analysis of each protocol.

Black Burst

One such MAC protocol was developed by Sobrinho and Krishnakumar [74] in Bell Laboratories. Their MAC protocol extends 802.11 to allow real-time traffic to co-exist with non real-time or data traffic.

In a similar approach to 802.11, the MAC protocol utilises three different inter-frame spacings between transmissions: t_{short} , t_{med} and t_{long} where $t_{short} + 2\tau \leq t_{med} + 2\tau \leq t_{long}$ where τ is the maximum propagation delay between pairs of hosts. t_{short} corresponds to the SIFS in 802.11, t_{long} corresponds to the DIFS and t_{med} corresponds to the PIFS (the PIFS of 802.11 is covered in section 2.2.3). Acknowledgement packets are sent after a t_{short} period in a similar way to the SIFS period for CTS and acknowledgement packets in 802.11. Real-time packets are transmitted by a mobile host after waiting for the medium to be idle for t_{med} time units. Finally, mobile hosts wait t_{long} time units when they have non real-time data to transmit.

As t_{med} is shorter than t_{long} , real-time packets can gain access to the wireless medium before non real-time. However if the wireless medium is idle for t_{med} time units and two mobile hosts have real-time packets to transmit then these two real-time packets would

collide if transmitted immediately. To prevent such collisions from occurring, Sobrinho and Krishnakumar designed a black burst contention period to determine whether a mobile host gains access to the wireless medium or not.

After the medium is idle for t_{med} time units, a mobile host with a real-time packet to send, transmits a sequence of energy bursts. The length of the sequence is proportional to the length of time that the mobile host has been waiting to access the medium. The length of each energy burst is chosen to allow a mobile host within range to detect the energy burst. After transmitting the energy burst, the mobile host listens for other mobile hosts transmitting a longer burst. If there is a longer burst, then the mobile host defers its transmission. Otherwise the mobile host transmits its packet. Thus the host that has been waiting the longest amount of time gains access to the medium to transmit their packet.

Mobile hosts with real time packets (bounded in length) are organised by the protocol into a TDMA-like structure provided that the transmission times of these real-time packets do not overlap. The designers of the protocol have derived a set of conditions for this TDMA-like structure to occur in the presence of data packets. If these conditions are met, then mobile hosts have time bounded access to the wireless medium.

However, even if these conditions are met, there is still a possibility of a mobile host's real-time packets colliding with other transmissions due to the hidden terminal problem. The protocol relies on every mobile host hearing the longest sequence of energy bursts. It is possible for a hidden terminal to not hear a longer energy burst of another mobile host and thus a collision of real-time packets will result. Sobrinho and Krishnakumar acknowledge this problem and introduce a negative acknowledgement scheme to request the retransmission of real-time packets that have collided. However, this negative acknowledge and retransmission highlights the fact that the previous time bound guarantee may not be obtained.

Splitting Protocol

Another approach to resolving contention when it occurs between mobile hosts was proposed by Markowski and Sethi [52] and uses a Contention Resolution Algorithm (CRA) to support the co-existence of hard, soft, and non real-time transmissions. An upper bound on the rate at which hard real-time packets are generated is assumed. The Markowski and Sethi MAC protocol also assumes that the mobile hosts in the network are all within communication range of each other and that the mobile hosts are uniquely numbered consecutively (from 0 to $n - 1$).

This MAC protocol is an example of a splitting protocol [85]. A mobile host transmits its packet (of a fixed size) and then listens for feedback information from other mobile hosts as to whether a collision has occurred or not. Collisions are detected by each mobile host that is listening. These listening mobile hosts give feedback to the transmitting mobile hosts by transmitting a short burst of energy (to indicate the occurrence or absence of a collision).

If a collision has occurred, then the CRA is executed. The CRA at each mobile host uses a window of size n (where n is the known number of hosts participating in the protocol) and divides this into an active part and an inactive part. A mobile host in the active window transmits while those in the inactive window defer. If another collision occurs, then the hosts in the active window are split again into an active and an inactive part. This procedure of window splitting continues until feedback indicates that no collision has occurred.

The worst case time for a hard real-time packet occurs when all n nodes transmit at the same time and then execute the CRA, this time is known as the Collision Resolution Interval (CRI). To provide real-time packets with a guarantee of successful transmission, only one real-time packet may be transmitted during any CRI period of time.

Analysis

The main disadvantage of the Sobrinho and Krishnakumar and the Markowski and Sethi medium access schemes is that they assume that participating mobile hosts are all within communication range of each other. Therefore, neither of these protocols deal with the well-known hidden terminal problem and are not suitable for multi-hop ad hoc networks.

Each of the protocols presented in the previous section resolve conflicts in a time bounded way. However, these time bounds may not be achieved in the presence of hidden terminals due to collisions.

In both protocols, mobile hosts are required to agree on important protocol related information before each protocol starts. The CRA requires the mobile hosts to agree on an ordering and numbering of the mobile hosts executing the protocol, this can either be done statically before the protocol starts or dynamically by a higher layer. In the energy burst protocol by Sobrinho and Krishnakumar, the packet size determines whether or not stability conditions are met and thus whether mobile hosts gain access to the medium in a time bounded way. No agreement protocol is described to allow a mobile host to dynamically vary the number of real-time packets that it wishes to transmit.

Another protocol by Facchinetti [25] makes similar assumptions to these protocols with regard to the hidden terminal problem (i.e., all hosts in the ad hoc network are fully connected).

2.2.3 Schedule-based Approaches

In this approach, mobile hosts negotiate with each other about when to transmit, so that these transmissions form a maximal set of collision free transmissions. This negotiation can be carried out in either a centralised or a distributed manner. Centralised

scheduling is typically used in infrastructure networks while distributed scheduling is more commonly used in ad hoc networks.

In a centralised approach, a host is assigned the task of determining which other mobile host within transmission range is allowed to transmit. In a distributed approach, mobile hosts agree on a set of TDMA slots in which to transmit, so that these transmissions are as free of collisions as possible. Once a mobile host has a slot allocated to it, it then has access to the medium until a new set of TDMA slots is negotiated. However, if two or more mobile hosts have inconsistent views of the allocated slots there is a possibility that these mobile hosts will corrupt each others' transmissions. These distributed approaches can be broken into two categories: topology-dependent and topology-transparent scheduling.

Centralised Scheduling

In this approach, a host is assigned the task of determining which other mobile host within transmission range is allowed to transmit. The host assigned this task is usually named differently depending on the particular wireless technology. For example, the host given this task is called a base station in GSM [63], an access point in the PCF of 802.11 [35] and a master in Bluetooth ([32], [54]).

Global System for Mobile communications

The first generation of mobile/cellular phones were analog. The Global System for Mobile communications (GSM) from the European Telecommunications Standards Institute (ETSI) [24] was the first fully digital system for both voice and data transmission. GSM operated originally in the 900MHz frequency band, this was later expanded to include both the 1800MHz and 1900MHz bands.

In GSM, a base station is assigned a geographical area called a cell. The base station coordinates access to the medium for mobile hosts located in the cell. A number of these

cells (and corresponding base stations) can be assigned to provide network coverage for a much larger geographical area. The base stations are joined together by some other, typically wired, communication network.

Within a cell, there is a maximum of 200 full-duplex channels. Each full-duplex channel consists of two frequencies. One frequency is for communication from the base station to a mobile host and one frequency for communication in the opposite direction. Each channel is divided into 8 time slots using time division multiplexing. The data rate of one of these slots is 9,600bps. Not all time slots or all frequencies are used in a cell in order to prevent transmissions in one cell from interfering with nearby cells.

Time slots are divided into data or control slots. Control slots are used for maintaining information about the current state of mobile hosts within a cell. A broadcast control slot is used as a continuous beacon from the base station in the cell identifying itself. Mobile hosts monitor this broadcast control slot to check when they have departed one cell and entered a new cell. Each base station maintains a database of the mobile hosts in its cell. A dedicated control slot is used to update this database and for slot management operations between mobile hosts and the base station. Finally, a random access slot is used by mobile hosts to request the base station to allocate the mobile host a dedicated control slot. As the name of this last slot suggests, there is a possibility of collisions if two or more mobile hosts transmit at the same time in this slot. If a collision does occur, then a mobile host uses a random backoff procedure to attempt to resolve this conflict.

802.11 Point Coordination Function

In contrast to the slotted approach that GSM takes, the Point Coordination Function (PCF) of 802.11 uses a master/slave approach to coordinating access to the wireless medium [84]. As mentioned in section 2.2.2, the PCF of 802.11 is designed for infrastructure networks. The master in the PCF is called an access point. The mobile hosts

that are under control of the access point and the access point together are called a basic service set (BSS). A BSS is similar to a cell in GSM.

Both the PCF and the DCF of 802.11 can co-exist by dividing access to the medium into two distinct periods. The Contention Free Period (CFP) is controlled by the access point and was designed for transmitting traffic such as voice and video (called synchronous data in 802.11). The second period is called the Contention Period (CP) and mobile hosts contend with each other using DCF to transmit their packets (called asynchronous data in 802.11).

At the start of each CFP, the access point waits for the medium to become idle for a PCF Inter-Frame Space (PIFS) time period and then transmits a beacon to initiate the CFP. As the size of the PIFS is shorter than the DIFS, the access point will gain access to the medium preempting any mobile host wishing to begin transmission. The access point immediately follows the beacon with one of four possible packets. One of these four packets, called CF-End, is used to end the CFP. The remaining three packets are used to transmit data to a mobile host, to poll a mobile host for data to transmit or to both transmit data and poll the mobile host. When a mobile host is polled for data to transmit, it responds within SIFS time units with either a no data acknowledgement or the data along with a CF-Ack packet. The mobile host could also choose to transmit a data packet to another mobile host in the BSS.

Access points can be connected to each other by some networking technology (either wired or wireless) called a Distribution Service in 802.11. However, the 802.11 standard does not make any recommendations as to which networking technology to use or what functionality should be placed at the access point for the distribution service.

Bluetooth

In contrast to the above infrastructure MAC protocols, Bluetooth is one of the few MAC protocols for ad hoc networks that uses a centralised approach. Bluetooth de-

vices operate in the 2.4GHz frequency band and when two (or more) devices establish communication, one device takes on the role of a master and the other(s) the role of slave. This newly formed master-slave network is called a piconet.

The 2.4GHz band is divided into 79 1MHz wide channels. A bluetooth device uses frequency hopping spread spectrum to hop between these 79 channels. A device hops between channels once every $625\mu s$ or 1,600 times per second. The frequency hopping sequence used in the piconet is calculated using the 48bit address of the master.

In a piconet, the master polls the slaves for any packets that they wish to transmit. A piconet can have a maximum of 7 active slaves and 255 parked slaves. Active and parked are two examples of a number of modes that a bluetooth device can enter during operation. These operation modes are designed to allow a bluetooth host to minimise the energy it uses by switching off its radio when it is not being used. A bluetooth device in the active mode listens for all transmissions from the master to remain synchronised. The master polls each of the active slaves in the piconet for packets to transmit. As a slave in active mode consumes a lot of energy by listening for each transmission from the master, a slave can switch to a less resource hungry mode. One such mode is called sniff mode, here a slave periodically becomes active and listens for transmissions from the master. Another power saving mode is the hold mode, here the slave stops listening to transmissions from the master for a specified period of time agreed between the master and the slave. Any slave in active, sniff or hold modes is considered part of the active slaves by the master. A slave in parked mode maintains synchronisation with the master using the slave's local clock. The master does not consider the slave as active when it is in parked mode.

When a bluetooth device powers on, the device goes into a standby mode. To transition from standby mode to communicating in a piconet, a bluetooth device invokes either an inquiry operation or a page operation.

An inquiry operation is invoked by a bluetooth device to learn the identity of

other devices within communication range. An inquiring device uses the “inquiring” hopping sequence and transmits an inquiry packet of duration $312.5\mu s$ during each hop. To successfully receive this inquiry packet, another device needs to be executing an inquiry scan which lasts for 1.28s on a particular frequency. Since the inquiring device changes its frequency much faster than the listening device, the probability of both mobile hosts being on the same frequency at the same time is greater.

During a page operation, one bluetooth device, typically the master, invites another device to join the piconet. The device issuing the invitation is called the paging device with the other device called the paged device. The paged device listens on a new frequency every 1.28s in a similar way to the inquiry scan. The paging device transmits a paging packet every $312.5\mu s$ using the page-hopping sequence.

Finally, it is possible for a device to participate simultaneously in more than one piconet. For example, a bluetooth device could be in hold mode in two piconets. As previously described, in hold mode the mobile host agrees with the master the duration of time that it will refrain from transmitting. During this time, the bluetooth device could switch its frequency hopping sequence to that of the second piconet and transmit its packets once polled by the master in the second piconet. A collection of two or more piconets with overlapping bluetooth devices is called a scatternet. It is possible for a master of a piconet to participate in a second piconet. However, a bluetooth device is restricted to acting as a master in only one piconet. The exact procedure for forming scatternets has not been specified in the current bluetooth standard versions.

Analysis

In terms of timeliness, each of the above MAC protocols provide time bounded communication to a certain set of mobile hosts in the network. The set of mobile hosts that have a bound on their transmission time are those that are known to the host that is controlling access to the medium. However, discovery of new mobile hosts is

not bounded in time. The underlying reason for this lack of a time bound is specific to the particular MAC protocol.

In GSM, recently arrived mobile hosts transmit a request to the base station on the random access channel to be allocated an available slot on an associated frequency. There is a possibility of collisions when two or more mobile hosts transmit on this channel at the same time. This is alleviated somewhat by base stations executing a handoff protocol to pre-allocate a slot to a mobile host when it leaves one cell and enters another cell.

The PCF of 802.11 is similar to GSM in that only those mobile hosts which are known to the access point are polled for packets to transmit. Newly arrived mobile hosts within communication range of the access point are required to transmit an association request during the CP to the access point to notify them of their presence. As transmissions during the CP use the DCF, again there is a possibility of collisions of these association requests.

In bluetooth, timely access to the wireless medium is provided to the master and active slaves in a piconet as the master polls the active slaves for transmissions. However, if a slave is in a parked state and there are 7 active slaves in the piconet, then the parked slave must wait until an active slave transitions to a parked state or it creates its own new piconet and requests a device in the old piconet to act as a gateway between the two piconets forming a scatternet. Thus, the duration for a parked slave to either become an active slave in the current piconet or to create a scatternet incorporating the old piconet could be unbounded. During startup, a bluetooth device can begin either an inquiry or page operation to join a piconet. In each of these operations, a device listens for approximately 1.28s for other devices transmitting, any worst case time bound will need to incorporate this listening duration.

In terms of agreement, neighbouring base stations in GSM agree on which frequencies that they will use to ensure that they do not interfere with each other. This

agreement is typically done statically when the network is being built. Similarly in 802.11, neighbouring access points need to agree to use different frequency hopping sequences in FHSS or different spreading codes in DSSS to prevent the possibility of transmissions in neighbouring cells interfering with each other. In bluetooth, devices agree on which device is to become the master. The master then coordinates slaves in the piconet, paging for new slaves, polling slaves for packets to transmit and allowing slaves to change modes from active mode to park mode, etc.

Multi-hop communication support has not been standardised in the bluetooth core specification. The formation of a scatternet depends on proprietary extensions by the manufacturer of the bluetooth device. In recent years, a number of techniques for scatternet formation have been proposed such as [70] and [59]. As previously mentioned, a limitation of scatternet formation is that a bluetooth device must be in hold mode when it is participating in two piconets. This would mean a corresponding increase in the delay that this bluetooth device has when accessing the medium.

The 802.11 standard does not specify how large scale infrastructure networks are constructed. The design and implementation of the distribution service is left to the manufacturer of the 802.11 device.

In contrast to the 802.11 standard, the GSM standard does specify protocols for the construction of multi-hop networks. For example, protocols are specified for the interaction between base stations. This allows seamless handoff of a mobile host that is moving from the cell of one base station to the cell of another base station.

Finally, as each of the above protocols rely on a central host to coordinate access to the wireless medium, the MAC protocol breaks down when this central host fails. In bluetooth, failure of the master would result in the slaves attempting to create a new piconet with one of the slaves as a new master. In GSM, mobile hosts would be unable to communicate if the base station failed and the mobile hosts were not within communication range of another base station. Similarly in the PCF of 802.11, if

the access point fails then the mobile hosts will be unable to communicate with other mobile hosts that were previously communicating with the failed access point.

HiperLAN [37] uses a central coordinator for determining access to the wireless medium. Peetz et al [36] specify an election algorithm that would allow HiperLAN to be used in ad hoc networks. This election algorithm is similar to the election of a master in bluetooth. In a similar way to bluetooth, time bounded communication in multi-hop networks is not achievable.

Topology Dependent Scheduling

Topology-dependent scheduling is a medium access strategy for ad hoc networks that assigns TDMA slots to mobile hosts (or links between mobile hosts) within a two-hop neighbourhood. As the topology changes, the slots are reassigned in a distributed manner by the mobile hosts.

One approach, from Cidon and Sidi [18], uses a dedicated set of slots as a control channel to resolve conflicts and broadcast channel reservations. The MAC protocol assumes that communication is bi-directional (i.e if mobile host A can receive transmissions from mobile host B then B can receive transmissions from A). Another assumption is that access to the medium is divided into a series of slots and there is a bound on the length of a packet so that it fits into one of these slots. The MAC protocol also assumes that every mobile host in the network is assigned a distinct priority (which can either remain fixed or can be changed deterministically by each host).

The goal of the protocol is to assign slots to hosts ready to transmit so that the number of successful transmission is maximal (i.e., any other transmission by any other host would result in a collision). This assignment may or may not be optimal, such an optimal assignment calculation has been shown to be an NP-complete problem [7].

The control channel, consisting of a dedicated set of slots, is proportional to the number of nodes in the network. Each host in the network is assigned two slots in the

control channel, a request slot and a confirmation slot. The request slots are at the beginning of the control channel followed by the confirmation slots. The confirmation slot is further sub-divided into two parts.

When a mobile host wishes to transmit, it transmits an indication of its intention to transmit in its request slot. Each mobile host listens for other indications from mobile hosts transmitting in the other request slots. During the following set of confirmation slots, if a mobile host wishes to transmit then it inhibits its neighbouring hosts by transmitting a deletion signal in the first part of each neighbour's confirmation slot and transmitting a confirmation signal in the second part of its confirmation slot. If a mobile host receives a deletion signal in the first part of its confirmation slot then it defers transmitting its packet. After the confirmation slots have passed, those mobile hosts that have not received a deletion signal in the first part of their confirmation slot, transmit their packet.

One problem of the above MAC protocol is that mobile hosts, with request slots at the beginning of the request segment, may gain access to the wireless medium more frequently than those at the end of the request segment. The designers of the MAC protocol address this issue using two different approaches. The first approach cycles the owners of the slots in the request segment in a round robin fashion. The second approach forces a mobile host that has been successful in transmitting a packet to defer subsequent transmissions until each of its ready to transmit neighbours has succeeded. The advantage of this second approach converges to a single TDMA cycle after a finite number of iterations.

Another topology-dependent approach, from Bao and Garcia-Luna-Aceves [9], called Neighborhood-aware Contention Resolution (NCR) uses local topology information to decide whether a mobile host can transmit or not. The NCR protocol assumes that every mobile host has knowledge of their one hop and two hop neighbours. A method to achieving this knowledge would be for each mobile host to periodically broadcast its

one hop neighbours. Another assumption of the NCR protocol is that the clocks of all the mobile hosts in the network are synchronised to within some known precision.

The NCR protocol operates by each mobile host generating a random number for each of its two hop neighbours. This random number is concatenated to the two hop neighbour's address to create a priority for this neighbour. Each mobile host uses the same pseudo random number generator. As this generator is seeded with the address of the two hop neighbour and the current contention round, the priorities generated for each two hop neighbour will be the same across all mobile hosts that carry out the calculation.

The priorities are sorted in descending order and the mobile host associated with the highest priority is allowed to transmit its packet. The NCR protocol can be augmented with a number of different channel access protocols. These different channel access protocols are designed for multicast/broadcast or unicast traffic with the unicast traffic utilising multiple DSSS codes to increase network efficiency.

The designers of the NCR protocol also describe a neighbour discovery protocol to allow mobile hosts to obtain knowledge about new one hop neighbours. This neighbour discovery protocol allows new mobile hosts to join the NCR protocol by notifying existing NCR participants of their presence. Access to the medium is divided between slots for the NCR protocol and slots for these joining mobile hosts. A joining mobile host randomly chooses one of these non-NCR slots and transmits a request to join the NCR protocol. Every existing mobile host within one hop of this new joining mobile host exchanges one hop neighbour information with this joining mobile host. After exchanging this information, a mobile host begins to take part in the NCR protocol.

Analysis

In the MAC protocol of Cidon and Sidi, the number of dedicated slots in the request and confirmation segments of the control channel is proportional to the number of hosts

in the network. Therefore, as the number of mobile hosts in the network increases, the number of dedicated slots also increases. This places a bound on the overall size of the ad hoc network, making this MAC protocol unsuitable for large multi-hop ad hoc networks.

Another problem occurs in the way that a mobile host reaches agreement with its neighbours on whether to transmit or not. A mobile host defers from transmission if it receives a deletion signal in the first part of its confirmation slot, otherwise the mobile host transmits. If the mobile host does not receive one or more of these deletion signals, then there is a possibility of collisions of data packets occurring.

In terms of timeliness, this MAC protocol allows mobile hosts to access the wireless medium within a known time bound. However, the size of this time bound will increase linearly with the number of hosts in the network.

In contrast, the NCR protocol was specifically designed for multi-hop ad hoc networks and would scale easily to large scale networks since it relies on local two-hop network topology information to execute. However, the NCR protocol (and the associated channel access protocols) have a number of problems. In the NCR protocol, whether or not a mobile host can transmit in the slot depends on the current state of the pseudo-random number generator and the mobile hosts local two-hop topology information. Therefore a mobile host is unable to know when it will gain access to the wireless medium again in a predictable way.

In addition, collisions can still occur using this approach when mobile hosts have an inconsistent view of the local one-hop and two-hop network topology (e.g., when the network begins to partition or merge). This would require that mobile hosts update their view of their two-hop neighbours in a consistent atomic way (see section 2.3) in the presence of mobile hosts joining or leaving the 2-hop neighbourhood. However, the neighbour discovery protocol is not an atomic agreement protocol. Some mobile hosts may be including the joining mobile host in their calculation of priorities of the NCR

protocol while other mobile hosts may not, possibly leading to collisions.

Topology Transparent Scheduling

In contrast to topology-dependent approaches, topology-transparent approaches allocate a collection of slots to a mobile host to use in an ad hoc network. The underlying idea is that if a mobile host transmits in each of its allocated slots then any neighbour of this mobile host will correctly receive at least one transmission from this mobile host. This is made possible by each mobile host using a unique code that determines which slots it will use.

Thus topology-transparent approaches are independent of the underlying network topology. However, these approaches do make some assumptions about the maximum number of mobile hosts in the ad hoc network and/or place a bound on the maximum number of neighbours that any mobile host can have. One limitation is that a transmitting mobile host does not know in which slot a particular neighbour can correctly receive their transmissions.

In the MAC protocol developed by Chlamtac and Farago [17], a unique code is generated using a distinct polynomial of degree k over a finite field (a Galois field $GF(q)$) [62] and is assigned to each mobile host in the network (here $q = p^m$ for some prime number p and some positive integer m). The values of q and k are chosen so that $q^{k+1} \geq N$ and $q \geq kD + 1$ where N is the number of mobile hosts in the network and D is the maximum number of neighbours that any mobile host in the ad hoc network can have. The solutions of a mobile host's distinct polynomial can have at most $k - 1$ roots in common with any other mobile hosts polynomial. The "graph" of each polynomial P over $GF(q)$ is defined by the protocol designers as the set of pairs $(\beta, P(\beta))$ where $\beta \in GF(q)$. Each mobile hosts polynomial graph is then mapped into a set of time slots for the mobile host to transmit in and thus a mobile host will have at least one slot $(k - (k - 1))$ to transmit in that is free of collisions from any of its two hop neighbours.

Using a similar approach, Ju and Li [38] use the Hamming distance [61] to generate a unique code to ensure that each mobile host has a minimum number of slots that allow successful packet transmission. Again a polynomial of degree k is assigned to each mobile host and this polynomial is mapped to a set of time slots. By using this approach, Ju and Li show that minimum throughput is maximised.

Another topology transparent protocol by Amouris [45] uses the position of a mobile host to determine in which slot that mobile host transmits. In this work, space is divided into virtual geographic cells called space slots in a similar way to cellular networking techniques. A slot is allocated to a cell and due to spatial reuse, this slot can be reused in cells sufficiently far away.

The approach by Amouris [45] could also be deemed to be topology dependent because if there is more than one mobile host in a cell, then access to the slot is negotiated among all the mobile hosts in that cell. The mobile hosts in a cell maintain a sorted list of mobile hosts in the cell and each mobile host on this sorted list transmits in the slot in a round-robin fashion. On entering a new cell, a mobile host broadcasts a packet to request other mobile hosts in the new cell to add the mobile host to the sorted list. This request in the new cell is also used to inform mobile hosts in the old cell of the departure of the mobile host. When another mobile host in the new cell receives this request, the receiving mobile host replies by transmitting a packet including the sorted list of mobile hosts located in the new cell.

A similar cellular approach is used by Caccamo et al [15] for time bounded transmission in wireless sensor networks. In contrast to the Amouris protocol, cells are assigned different frequencies to allow simultaneous transmissions in each cell. As this MAC protocol is designed for sensor networks, an assumption is that hosts in the network are static and that real-time or time bounded messages are periodic. The MAC protocol uses a schedule for packet transmission (the schedule is based on earliest deadline first). This schedule is replicated at each of the hosts in the cell. The MAC protocol also uses

routing hosts to enable communication between cells.

Analysis

As highlighted in the previous section, one limitation of the topology transparent MAC protocols, by Chlamtac and Farago, and Ju and Li, is that a mobile host does not know in which slot a particular neighbour can correctly receive their transmissions. Therefore, a mobile host must transmit in each of its allocated slots to guarantee successful transmission to its neighbour. Note that even if a mobile host succeeds in transmitting in a particular slot, this does not imply that this slot will be collision free for each subsequent transmission. Whether a slot allocated to a mobile host is collision free depends on the slots allocated to the neighbours of this host. As the neighbours of a mobile host can change due to mobility, the slots that are free of collisions also change. In terms of timeliness, the worst case delay for a mobile host to transmit is twice the frame length (the frame length is the duration of time it takes for all the allocated slots of a mobile host and its two hop neighbours to pass).

Another limitation of these two MAC protocols is that they place a bound on the number of neighbours that any mobile host in the ad hoc network can have. There is also an upper bound placed on the number of mobile hosts in the network. Mobile hosts also need to be statically assigned a code that is unique to them before transmissions on the network begin.

There are a number of problems with the protocol developed by Amouris in terms of a mobile host having predictable access to the wireless medium. Firstly, the author of the paper does not specify how a mobile host that powers on in a cell obtains a slot. A similar problem arises when two mobile hosts enter an empty cell or power on in an empty cell at the same time. Another problem is that the sorted list of mobile hosts is replicated by each mobile host in the cell. Therefore, updates to this replicated data structure need to be carried out in a consistent manner, otherwise there is a possibility

of the sorted list becoming inconsistent across mobile hosts, which eventually results in collisions in the assigned slot.

The protocol by Caccamo also has a number of problems. Firstly, the hosts/sensors in the cell are assumed to be static. The schedule for transmissions used by these hosts within a cell is also static. Updating this schedule dynamically would require some form of atomic agreement between the hosts in the cell. Another problem is that the existence of a routing host is assumed to be located at the center of each cell. How this routing host is elected and what happens when such a host fails is a topic of future research by the protocol designers.

2.3 Agreement Protocols

In this section, a number of important protocols for reaching distributed agreement among a group of hosts (processes) are discussed. Firstly, background information relevant to the rest of this section is discussed. Following this discussion, two different types of approaches to allow hosts to reach distributed agreement are covered.

2.3.1 Background

Over the last three decades, a number of protocols have been developed to address a number of problems in distributed systems. One of the main problems that has been addressed is how to allow data to be replicated across a number of hosts/processes consistently.

The problem of maintaining replicated data introduces the distributed commit problem [31]. This problem manifests itself when a collection of hosts wish to perform some action in a consistent way. Either all the hosts perform the action or none do.

As there are a number of hosts and these hosts are connected by some communication medium, there are a number of different types of failures affecting these protocols

[21]. Example failures include crash failure of one or more participant hosts, omission failures of the medium and arbitrary or byzantine failures of one or more hosts.

Another fundamental problem in distributed systems research has been how to detect a failure (usually of a host/process in a group) when it occurs. Typically, this is done by a group membership service monitoring the members of the group.

Models of group membership can be either static or dynamic. The current membership of the group is usually called the current group view. In the static group membership model, there is a maximal set of hosts (or processes) that can be members of the group. The actual group members comes from this static set and the group membership varies dynamically over this set depending on the number of failures and recoveries of the group members. Each process typically has a unique identifier; when a process fails and restarts, the process reuses this unique identifier.

In the dynamic group membership model, processes join the group and transmit/receive messages to/from the group before finishing. The membership at a given point in time is defined as the set of processes that are currently operational in the system. Each process has a unique identifier. Unlike the static model, when a process fails and rejoins, the process does not reuse its previous identifier.

In both the static and dynamic membership models, a problem arises when starting the group, where initial group members need to discover each others existence. Typically this problem is solved by statically configuring an initial membership (consisting of a small number of processes or hosts) which is then started. This allows the group members to have a consistent view of each others presence. In the static model, the remaining members then join. In the dynamic model, the membership changes as new processes are created and old processes either fail or leave.

Given a set of members of a group, there are a number of different properties related to how group members transmit messages to the group. One such property is whether the transmission of a message is atomic or not. An atomic message reaches all the

operational group members or none. Another property of a message transmitted to a group is whether or not the delivery of the message is ordered with respect to the delivery of other messages transmitted to the group. There are three different ways of ordering messages sent to a group: FIFO ordering, causal ordering and total ordering.

In FIFO ordering, messages from a sender are delivered to the group in the order that they were transmitted by that sender. A fifo ordered message can be either atomic or not.

Causal ordering can be thought of as an extension to FIFO ordering where the sending of a message can be thought of as having “happened before” the reception of the same message. By modelling the transmission and reception of messages as events, it is then possible to define a partial ordering on these events. This partial ordering relation can then be extended to a causal ordering on events in the group by taking its transitive closure.

This causal ordering relation is usually denoted $\rightarrow$. For two events, A and B, either $A \rightarrow B$ or $B \rightarrow A$ or A and B are deemed to be concurrent. Here, $A \rightarrow B$ means that event A happened before event B. Concurrent events can be delivered in any order at the members of the group.

Totally ordered messages sent to a group are delivered by each member of the group in the same order. This contrasts with causal order where concurrent messages can be delivered in any order at group members. Typically, totally ordered messages to a group are atomic.

2.3.2 Rounds of Message Passing

In these approaches, processes exchange messages in a series of rounds to maintain replicated data, achieve atomicity or provide some form of ordering. One of the first such techniques was the Two-Phase Commit protocol designed to maintain replicated data. The Two-Phase Commit protocol allows a static set of hosts to agree on whether

or not to perform an update on some shared data.

The Two-Phase Commit protocol works by the host proposing the update acting as a coordinator for the protocol sending out a message to the other hosts asking them if they are prepared to “commit” the update to the replicated data. Recipients of the message respond to the coordinator indicating whether they are prepared to commit or abort the update. The coordinator waits for each of the recipient hosts to respond and then collates the recipients decision. If all the recipients are prepared to commit the update, then the coordinator sends out a commit decision otherwise the coordinator sends out an abort decision to the recipients.

The Two-Phase Commit protocol suffers from a number of problems when failures occur [11]. In the worst case where the coordinator is also a participant in the protocol, if the coordinator fails the protocol can be blocked until the coordinator recovers. To address some of the problems of the Two-Phase Commit protocol, Skeen and Stonebraker [30] modified the Two-Phase Commit protocol to include an extra round of message passing between the coordinator and the recipients. This new protocol, called Three-Phase Commit, can also block in the presence of network partitions.

Other protocols designed to allow updates to replicated data include quorum techniques where updates can be performed with less than the full set of static hosts participating. Two concurrent updates are carried out so that there is at least one overlapping replica at which both updates are being performed.

A number of protocols have used message passing to provide atomicity and total ordering. For example, Chang and Maxemchuk [16] use a token passing scheme to order messages before delivery. The host that currently holds the token decides on an ordering for the messages transmitted before passing the token to the next host. Another approach to ensure total ordering is to elect a dedicated host as a sequencer to order the transmitted messages [39]. On failure of the sequencer, the remaining hosts elect a new sequencer.

A number of researchers have investigated different ordering protocols in the virtual synchrony model [12] [8] for dynamic group membership. The goal of the virtual synchrony model is to provide support for the replication of data across a number of processes.

One of the properties of virtual synchrony, called view synchronous communication, is that any message sent to a particular group view will be delivered before a new group view is installed. FIFO, causal and total ordering protocols have been developed using this view synchronous communication property [13].

2.3.3 Diffusion Protocols

In contrast to protocols using rounds of message passing, a number of protocols have been developed using diffusion of messages to the protocol participants. In a diffusion protocol, a message is flooded throughout the network to all the hosts participating in the protocol.

A collection of diffusion protocols were developed by Flaviu Cristian et al. for a static set of hosts [22], [20]. The protocols developed by Cristian used synchronised clocks and provided atomic totally ordered broadcast that tolerated a known number of failures. The protocols assumes that there is a total order on the set of hosts and that the hosts are connected by a number of replicated networks (also ordered).

When a host wishes to send a message, it sends the message along with a delivery time for the message, on all replicated networks in increasing order. Upon reception of a message, a host re-transmits the message if the message was not seen before and has not passed its delivery time. When the delivery time of the message arrives, the message is passed to the application. The delivery time is chosen so that the message will reach all hosts in the presence of a known maximum number of failures. If two or more messages are scheduled for delivery at the same time, then they are delivered in a consistent order according to the total order on the set of hosts.

The diffusion protocol just described can be considered as an example of a gossip protocol [23]. There are two types of gossip protocols, push and pull. In push gossip protocols, when a process receives a message, the process randomly chooses a subset of processes from the set of participating processes. The process then forwards the message to the chosen subset of processes. The size of the subset chosen in relation to the number of participating processes is equivalent to the probability that a process retransmits a received message. In the Cristian protocols, the probability of a host retransmitting is equal to 1 (i.e., a host always retransmits).

In contrast to push gossip protocols, a process using a pull gossip protocol periodically contacts a random subset of the participating processes and queries the subset for any messages that they have.

The advantage of gossip protocols is that the number of message retransmissions to reach all the participating processes is logarithmic in relation to the number of participating processes [11]. Another advantage is that gossip protocols do not rely on a specific process to carry out a specific function. Thus gossip protocols have the advantage of continuing to operate during the failure of any process. Gossip protocols have also been labelled epidemic algorithms to reflect the similarity between the flooding of messages and the spread of disease in a biological population.

The Trans protocol [53] can be considered as a hybrid protocol using both diffusion and message rounds to build a reliable broadcast protocol. In the Trans protocol, hosts share a broadcast medium called the broadcast domain. The protocol works by participants positively acknowledging messages that they have received and negatively acknowledging those messages that they have not received correctly. By using these positive and negative acknowledgements, both a partial and a causal order have been developed.

The Transis project [51] used this Trans protocol to build a virtual synchrony based group communication service that provided total ordering. Transis also extended the

Trans protocol to work over multiple hops.

2.4 Summary of Related Work

At the beginning of this chapter, some of the important properties, such as timeliness, agreement and network topology, provided by time bounded MAC protocols are covered. As seen in section 2.2, the MAC protocols for ad hoc networks do not provide all of these properties.

Some of the MAC protocols described have focussed on maximising the overall network throughput and do not provide timely access to the wireless medium. Other MAC protocols do provide time-bounded access to the wireless medium but each of these protocols do not address one of two issues. Firstly, some of these latter MAC protocols do not address the hidden terminal problem and assume full connectedness of the protocol participants. Secondly, in the MAC protocols that do address the hidden terminal problem, collisions can still potentially occur due to MAC protocol participants having inconsistent information about the current state of other protocol participants.

An approach to solve this last problem of inconsistent information would be to run some agreement protocol between participants of the MAC protocol. Such an agreement protocol would need to know the participants (or group members) that are attempting to reach agreement. However, an assumption of both the static and dynamic membership models covered in section 2.3, typically, is that an initial static set of group members is started that know about each other's presence. This is an unrealistic assumption in multi-hop ad hoc networks where a mobile host needs to discover the neighbouring hosts within its communication range when it powers on and the number of neighbouring hosts can change due to mobility.

Table 2.1 summarises whether the MAC protocols covered in this chapter provide

ad-hoc support, reach agreement between the protocol participants and address the issues of timeliness.

Approach	Protocol	Ad Hoc	Agreement	Time bounds
Schedule based				
Centralised				
	GSM	×	✓	✓
	802.11 PCF	×	×	✓
	Bluetooth	single-hop	×	×
Distributed				
	Cidon and Sidi	multi-hop	×	×
	NCR	multi-hop	×	×
	Chlamtac and Farago	multi-hop	×	×
	Ju and Li	multi-hop	×	×
	Amouris	multi-hop	×	×
Contention based				
Reducing Contention				
	MACA	multi-hop	×	×
	MACAW	multi-hop	×	×
	802.11	multi-hop	×	×
Resolving Contention				
	Black Burst	single-hop	×	✓
	Splitting	single-hop	×	✓

Table 2.1: Reviewed MAC protocol properties

Chapter 3

A New MAC Protocol for Ad Hoc Networks

This chapter describes the Time Bounded Medium Access Control (TBMAC) protocol for multi-hop ad hoc networks. The goal of the TBMAC protocol is to provide, with high probability, mobile hosts with time-bounded access to the wireless medium. TBMAC provides a set of time bounds on gaining access to the wireless medium that depend on the current operational state of the protocol.

Section 3.1 introduces the basics of the TBMAC protocol. As highlighted in chapter 1, the TBMAC protocol uses time division multiple access with dynamic slot allocation. Dynamic slot allocation is achieved through the use of an atomic multicast protocol that is described in section 3.2. Following this, section 3.3 gives a detailed description of the TBMAC protocol. Finally section 3.4 gives the time bounds for a mobile host having predictable access to the wireless medium and derives the corresponding probabilities of these time bounds being obtained.

Note that the TBMAC protocol provides the functionality to allow mobile hosts to gain predictable access to the wireless medium and to manage the allocation and deallocation of TBMAC slots to mobile hosts. However, whether a mobile host should

be allocated three TBMAC slots instead of two TBMAC slots is a decision left to layers above the TBMAC protocol. TBMAC is not concerned with the best or optimal way to manage the allocation of TBMAC slots to a set of mobile hosts. During the course of this chapter, the issue of the management of the allocation of TBMAC slots and the functionality provided by the TBMAC protocol to enable this management will be highlighted.

3.1 Protocol Basics

The TBMAC protocol imposes a number of requirements on the participants of the protocol in the ad hoc network. These requirements must be satisfied for the TBMAC protocol to function. Once the requirements are satisfied, the TBMAC protocol provides the protocol participants with a set of time bounds (with associated probabilities) on gaining access to the wireless medium. The next sections discuss these requirements and introduces the obtainable time bounds.

3.1.1 Cellular Structure

To reduce the probability of the transmissions colliding, the geographical area occupied by the mobile hosts is statically divided into a number of geographical cells in a similar approach to [45]. Each cell can have arbitrary shape and size but for simplicity of presentation, we assume that the cells are hexagons of equal size as illustrated in Figure 3.1. Each cell is numbered and, in contrast to [45], is allocated a unique radio channel (or CDMA spreading code).

The motivation behind dividing the area of coverage into a collection of fixed size cells is twofold. Firstly, it reduces the possibility of the well-known hidden terminal problem [81] from occurring by reducing the possibility of interference from mobile hosts in neighbouring cells. Reducing the likelihood of interference results in the probability

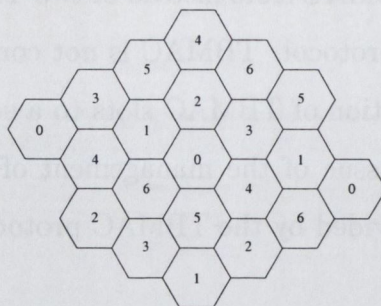


Fig. 3.1: Possible Cell and Channel allocation

of one mobile host in a cell correctly receiving the transmission of any other mobile host in the same cell [64]. In order to reduce the likelihood of the hidden terminal problem occurring, the width of a cell is related to the transmission range of the wireless technology being used.

Secondly, dividing the area into a collection of cells with corresponding radio channels increases the total overall bandwidth available in the ad hoc network by reusing radio channels in cells sufficiently far away. For example, this spatial reuse strategy [64] allows the transmissions of mobile hosts in the rightmost “0” cell in figure 3.1 to occur simultaneously with transmissions of mobile hosts in the center “0” cell.

The assumption of a cell of fixed size is similar to the single hop ad hoc network assumption of the wireless real-time MAC protocols covered in section 2.2.2. In addition to enabling communication within a single cell, another goal of the TBMAC protocol is to allow multi-hop communication to take place which corresponds to communication between adjoining cells.

Decreasing the width of a cell without modifying the underlying wireless technology (i.e., keeping the transmitting power used constant) should result in a corresponding increase in the probability of correct packet reception between any two mobile hosts in that cell. However, decreasing the width of the cell impacts on the spatial reuse strategy being used.

As previously mentioned, the layout of the geographical cells is static. The mobile hosts in the ad hoc network are required to know this layout. This includes each mobile host knowing the boundaries of each of these cells. To meet this requirement, each mobile host requires access to location information (such as GPS [43], [82]). When a mobile host knows which cell it is in, it can infer the correct radio channel to use.

We assume the presence of a layer above the TBMAC that decides in which cell the mobile host is currently located and therefore which radio channel should be used. The higher layer would continually monitor the current position of the mobile host (based on the update rate of a GPS receiver for example) and notify the TBMAC protocol of a change of cells. The operation of such a higher layer will not be discussed further in this chapter.

3.1.2 Periods of Operation

In the chapter 2, the advantages and disadvantages of the TDMA MAC protocol were covered. The main advantage of TDMA is that hosts have predictable access to the medium. One of the disadvantages of TDMA is that hosts are statically allocated the TDMA slot to use before the TDMA protocol begins. One view of the TBMAC protocol is that it attempts to provide the predictable access of TDMA to mobile hosts but removes the requirement for slots to be statically allocated to mobile hosts. Instead of statically allocating slots to mobile hosts, the mobile hosts dynamically reach agreement on updates to the allocation of slots using the agreement protocol described in section 3.2.

To provide both predictable access to mobile hosts and allow mobile hosts to dynamically request to be allocated a slot, the TBMAC protocol divides access to the wireless medium within a cell into two distinct time periods:

1. Contention Free Period (CFP)

2. Contention Period (CP)

Both the CFP and the CP are divided into slots and each period lasts a predefined period of time. A CFP followed by a CP constitute a round of the TBMAC protocol. Typically, slots in the CP are shorter in duration than CFP slots since they will usually be used to transmit less information than CFP slots. Once a mobile host has been allocated a CFP slot, it has predictable access to the wireless medium. The mobile host can then transmit data in its slot until it leaves the cell or fails.

Mobile hosts that do not have CFP slots allocated to them contend with each other to request CFP slots to be allocated to them in the CP. The CP is used by mobile hosts that arrive into the cell or that have recently powered on in the cell. The steps carried out by a mobile host to be allocated a CFP slot are covered in section 3.3.

Dividing access to the medium into these two well-known time periods requires the clocks of all the mobile hosts in the network to be synchronised. Again equipping each mobile host with a GPS receiver would satisfy this clock synchronisation requirement [83] [27]. Thus, in each cell at approximately the same time (based on the precision of the clocks of mobile hosts in the network), each slot in the CFP begins and ends followed by each slot in the CP.

To enable communication between adjoining cells, one or more slots in the CFP, called inter-cell slots, are statically allocated. In a similar way to the configuration of geographical cells, the mobile hosts in the ad hoc network are required to reach agreement on these static inter-cell slots before the TBMAC protocol starts executing. Mobile hosts must then dynamically agree on which mobile hosts are allocated each of these inter-cell slots to transmit in.

3.1.3 Time Bounds

With the geographical cell structure described above, the CFP in each cell can be in two possible states. Firstly, a set of mobile hosts can be in a cell and have agreed on

a consistent allocation of CFP slots (note that we are implicitly excluding the case of mobile hosts trivially agreeing on an allocation of CFP slots, e.g., all CFP slots allocated to a default mobile host). In TBMAC terminology, this state is called a non-empty CFP; the cell is said to be a non-empty cell and the protocol is in non-empty or operational mode. Secondly, mobile hosts may not have agreed on a consistent allocation of CFP slots in a particular cell. This state is called an empty CFP, the cell is called an empty cell and the protocol is in bootstrapping mode.

In each of these types of TBMAC modes, empty or non-empty, there are a number of time bounds. It is important to note that these time bounds provided by the TBMAC protocol are not worst case time bounds which are traditional in the hard real-time community. To provide such worst case time bounds would require, among other things, an upper bound on the number of transient errors that could occur in a well-defined period of time. This error information would be very specific to the environment in which a particular MAC protocol is executing. The TBMAC protocol takes a different approach by providing a set of time bounds and associated to each of these time bounds is a corresponding probability of these time bounds being obtained. Layers above the TBMAC protocol can then be tailor these time bounds (and probabilities) to their particular needs.

As mentioned in the previous section, in the non-empty cell case, once a mobile host in a cell has been allocated a CFP slot, the mobile host has access to the medium within one round of the TBMAC protocol (i.e., the duration of the CFP plus the duration of the CP denoted $(CFP + CP)$).

Again in a non-empty cell, if a mobile host with an allocated CFP slot wishes to allocate another CFP slot, then the mobile host executes the agreement protocol using its existing CFP slot to reach agreement with the other mobile hosts that have CFP slots allocated in the cell on the allocation of the new CFP slot. The duration of time needed to execute this agreement protocol is equal to two rounds of the TBMAC

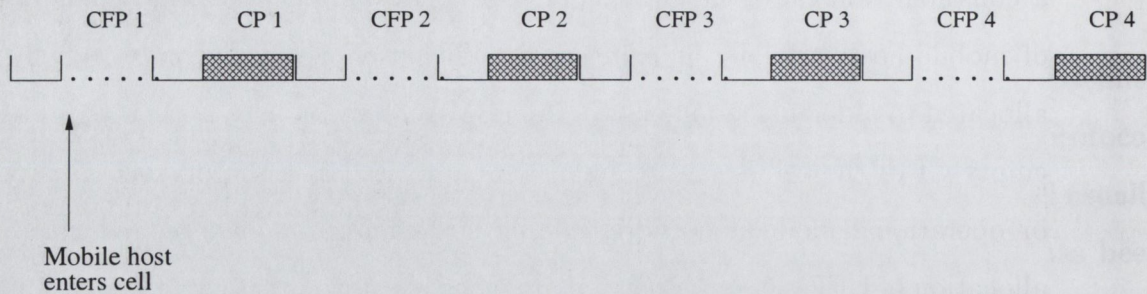


Fig. 3.2: Mobile host entering a cell in the CFP

protocol $2 * (CFP + CP)$ (see section 3.2). After this agreement protocol is executed, the mobile host can transmit in this newly allocated slot within one round of the TBMAC protocol. Thus, the time bound before accessing this newly allocated CFP slot is $3 * (CFP + CP)$.

When a mobile host arrives or powers on in a cell, as illustrated in figure 3.2, and wishes to be allocated a CFP slot, the mobile host first waits for one full CFP to pass (CFP 2 in the figure). Waiting this period of time allows the mobile host to detect whether it is in a non-empty cell or an empty cell based on whether or not it has received any transmissions in the CFP.

In a non-empty cell, the mobile host then transmits a request in the following CP (CP 2 in the figure) to be allocated a CFP slot. The transmission of this request may collide with other contending mobile hosts that are also requesting a CFP slot. However, the probability of this collision occurring can be calculated. If a mobile host with an allocated CFP slot correctly receives this request, this mobile host executes the agreement protocol on behalf of the requesting mobile host (taking 2 TBMAC rounds, CFP 3 and 4 in figure 3.2).

The time bound in this case occurs when a mobile host enters the non-empty cell just as the CFP begins, the mobile host must wait for the end of this CFP before listening to the next CFP, request a CFP slot in the following CP and then wait

for the agreement protocol to be executed before beginning to transmit in the newly allocated slot (in CFP 5 not in the figure). Thus, this time bound is $5 * (CFP + CP)$.

In an empty cell, the mobile host executes a bootstrapping protocol to allocate the first CFP slot in the cell (see section 3.3.4). It is possible that more than one mobile host will execute this bootstrapping protocol at the same time. Thus the bootstrapping protocol must allow two or more mobile hosts to detect each others presence in the cell and then agree on a consistent allocation of CFP slots. In addition, the bootstrapping protocol should not prohibit a single mobile host from allocating one or more CFP slots.

To have an associated time bound in the empty cell case, the execution of the bootstrapping protocol must be limited to a fixed and known number of TBMAC rounds, M say. At the end of this bootstrapping protocol, the mobile hosts executing the protocol switch to the non-empty TBMAC mode with each mobile host in one of two possible states.

Firstly, the mobile host can have one or more CFP slots allocated to it that are consistent with the other mobile hosts executing the bootstrapping protocol. The time bound for such a mobile host is $(2 + M) * (CFP + CP)$.

Secondly, there is a possibility that the mobile hosts executing the bootstrapping protocol fail to reach agreement on a consistent allocation of CFP slots at the end of the bootstrapping period. In this case, some or all of the mobile hosts will mistakenly believe that their inconsistent allocation is consistent and continue to execute the TBMAC protocol. This would result in collisions occurring in the CFP during non-empty or operational mode.

However, it is possible to calculate an upper bound on the probability of this inconsistency occurring and thus a corresponding lower bound on the probability of the above empty cell time bound being obtained (see section 3.4).

Finally, if a mobile host enters or powers on in the cell while the bootstrapping

protocol is being executed, then this mobile host must wait until the end of the bootstrapping protocol before requesting a CFP slot to be allocated to it. The time bound for such a mobile host is $(2 + M) * (CFP + CP)$.

In the above discussion, the calculation of these probabilities is predicated on a model of the arrival rates of mobile hosts into a cell being provided.

3.2 Agreement Protocol

The requirement for distributed agreement to enable coordination/cooperation between mobile hosts in an ad hoc network and, in particular, time bounded or predictable medium access control was briefly outlined in Chapter 1. A number of possible problems were highlighted for some of the MAC protocols covered in Chapter 2 if distributed agreement is not used to update replicated MAC protocol information. One requirement of the TBMAC protocol is to update in a consistent way the allocation of CFP slots at all the mobile hosts in a particular cell.

The division into a CFP and a CP in TBMAC is similar to the Point Coordination Function (PCF) in the 802.11 standard (see section 2.2.3) with, as we shall see, the exception that the TBMAC CFP does not rely on one particular mobile host to act as an access point. In the PCF of 802.11, agreement on accessing the wireless medium is enforced by the access point polling the mobile hosts for data to send. In the ad hoc environment considered here, we cannot assume there is an access point present in the area covered by the geographical cell.

One option would be to elect an access point from the mobile hosts present in the cell. This option has a number of problems.

Firstly, to provide predictable medium access to the mobile hosts, these mobile hosts have to reach a distributed agreement within bounded time on which mobile host is to become the access point. To reach such an agreement between the mobile hosts within a

known time bound without the hosts having predictable access to the wireless medium is difficult to envisage. Thus, using the distributed election approach, the problem of providing predictable access to the medium depends on an election algorithm that executes in bounded time which in turn depends on mobile hosts having predictable access to the medium.

Secondly, if it was possible to elect such an access point in bounded time, then the mobile hosts would also have to monitor the access point for failure and to reach agreement in bounded time that such a failure has occurred and then to instigate the access point election algorithm again. In summary, the idea of electing an access point simply moves the problem of reaching distributed agreement on access to the wireless medium to distributed leadership election and failure detection.

Instead of using an access point, the TBMAC participants reach a distributed agreement on the order of allocations and deallocation of CFP slots. Updates to the allocation of CFP slots in a cell need to be totally ordered across all mobile hosts in the cell. Two or more messages (e.g., allocating CFP slots) are seen in the same order by every mobile host in the cell.

The approach we use to provide such a total ordering protocol within a cell is to use the synchronous atomic multicast protocol from Flaviu Cristian [20]. Cristian's protocol is designed to totally order a sequence of messages transmitted using the protocol in the presence of a known number of failures occurring. Typical uses of this protocol in the TBMAC protocol are to allocate and de-allocate slots in the CFP and for requests for inter-cell slots to communicate across cell boundaries.

To explain how the synchronous atomic broadcast protocol works, consider a mobile host in a cell with a CFP slot already allocated to it that wishes another CFP slot to be allocated to it. We will defer the description of the procedure for allocating the first CFP slot in a cell to section 3.3.4.

The mobile host creates a packet requesting a slot, including information related

to the atomic broadcast such as the mobile host's address, a sequence number and the current time (a timestamp). The mobile host then broadcasts this packet a number of times using its CFP slot until the delivery time of the message arrives. The delivery time of the message is equal to the original timestamp plus the delay to delivery, Δ , which is a parameter of the atomic broadcast protocol.

When a mobile host receives such a packet that it has not received, it stores it based on the address and the sequence number in the packet. In addition to storing the packet, the mobile host also rebroadcasts the packet in each of its CFP slots until the delivery time of the information in the packet arrives.

The logic behind the originator of a message broadcasting a message a number of times is to reduce the probability of one or more errors or failures occurring during the transmission of the message. A receiver rebroadcasts the message to overcome the possibility of the failure of the original sender of the message and of the possibility of errors during transmission. Thus, atomicity, up to a fixed number of failures, is provided by the sender of the message and the receivers of the message broadcasting the message a number of times until the delivery time of the message arrives.

The delivery delay, Δ , for the TBMAC protocol would typically be 2 rounds of the TBMAC protocol ($2 * (CFP + CP)$). This would allow the TBMAC protocol to tolerate 1 failure when reaching agreement using the Cristian protocol.

When the delivery time arrives, all the mobile hosts in the cell then update their slot allocation information consistently to incorporate the new CFP slot of the requesting mobile host. Consistency in the ordering of the updates to the allocation of slots is ensured by each mobile host ordering two or more messages with the same delivery time in the same way. Messages with identical delivery times are ordered first by the address in the message and second by the message's sequence number. This ensures the total ordering property of the Cristian protocol.

3.3 Protocol Details

In this section, we give an in depth description of the TBMAC protocol. Firstly, the data structures and packet types used by TBMAC are covered. Following this, section 3.3.3 covers the steps involved in allocating and deallocating a CFP slot in a non-empty cell. Following this, the procedure to allocate the first CFP slot to a mobile host when it enters an empty cell is discussed. Finally in this section, a description is given of some possible extensions to the TBMAC protocol.

3.3.1 Data Structures

There are two important data structures maintained by each mobile host running the TBMAC protocol, these are the slot owners and slot bitmap structures. Both the slot owners and slot bitmap are data structures that are replicated across the mobile hosts in the cell. These replicated data structures are updated consistently using the atomic multicast protocol described in the previous section.

Slot owners is an array of addresses whose size is equal to the number of slots in CFP, which stores the address of each mobile host that has a CFP slot allocated to it. The slot bitmap contains two bits for each slot in the CFP to represent the four possible states of a slot: Owner, Other, Collision, Available.

When a mobile host sets a position in the slot bitmap to Owner, it is indicating to other mobile hosts that it is using this slot. Similarly, by setting a slot position to Other, the mobile host is indicating that another mobile host is using that slot. The identity of the mobile host that is using the slot is stored in the mobile host's slot owners array. A CFP slot is marked Available in the slot bitmap if the mobile host is not aware of any other mobile host using that slot. Finally, a CFP slot is marked in the slot bitmap as Collision when a mobile host has received a message (in another CFP slot) indicating that a collision is occurring in this slot (see section 3.3.4).

Source Address	Destination Address	Allocated Slot Bitmap	Current Slot Number	Message Type	Additional Info	Protocol Extensions
-------------------	------------------------	--------------------------	------------------------	-----------------	--------------------	------------------------

Fig. 3.3: CFP Slot Header

When a mobile host transmits in a CFP slot, it includes its slot bitmap. A similar technique is used in the Time Triggered Protocol (TTP) described in section 2.1.2. However, the motivation for including this slot bitmap in each transmission is different in TBMAC than in TTP. In TTP, the slot bitmap is used to detect the failure of a host or hosts to transmit in the slot allocated to them. In contrast to TTP, TBMAC uses the slot bitmap to represent the current state of each of the CFP slots in a particular cell. An additional difference is that the slot bitmap in TTP represents a set of binary values while one position in the slot bitmap in TBMAC represents four possible values.

3.3.2 Protocol Header

When a CFP slot is allocated to a mobile host in a cell, the mobile host includes the CFP header (Figure 3.3) in each of its transmissions in its CFP slot. As previously mentioned, a mobile host includes its Slot Bitmap as part of this CFP header. A mobile host also includes the source and destination address and the current CFP slot number in the header.

The Message Type field is used to indicate the type of message being transmitted. The different possible types of messages are:

1. Data
2. Null

The need for the first message type is obvious and is used when a higher layer above TBMAC wishes to transmit data in its CFP slot. Note that TBMAC does not specify

how packets from higher layers should be scheduled for transmission. The solution to this scheduling problem is the responsibility of a higher slot management layer.

When a CFP slot is allocated to a mobile host, the mobile host sends a Null message in its slot even if it does not have a message to send. The transmission of a message (data message or Null message) in each CFP slot allows an arriving mobile host to learn the current allocation of CFP slots through the reception of one message.

Following the message type field, the Additional Info field is a variable length field (up to a known maximum) and its use depends on the particular mode that the TBMAC protocol is currently in.

When in a empty cell, the Additional Info field contains information related to the bootstrapping protocol (see section 3.3.4).

When in a non-empty cell, the Additional Info field contains updates to the allocation of CFP slots that are currently ongoing using the agreement protocol. The possible updates include:

1. Slot Allocation Request
2. Slot Deallocation Request
3. Leave
4. Inter-Cell Communication Request
5. Slot Failure

Briefly, the Slot Allocation and Deallocation Request updates are used by mobile hosts to allocate and deallocate CFP slots. The Leave update is used by a mobile host to indicate its intention to leave the cell. The Inter-Cell Communication Request update is used when a mobile host wishes to communicate across its current cell boundary with a neighbouring cell. This request is further subdivided based on whether a CFP slot is requested for communication from the current cell to the neighbouring cell, from

the neighbouring cell to the current cell or for communication in both directions (see section 3.3.3). The Slot Failure update is used when one mobile host suspects another mobile host of having failed (see section 3.3.3).

On first reading of the description of agreement protocol in section 3.2, it would appear that if a mobile host wishes to reach agreement on the allocation of a slot with the other mobile hosts in a cell then it needs to retransmit the same message a number of times in its slot. However, since the updates transmitted using the atomic multicast protocol are relatively small, this information can be inserted into the Additional Info field and thus be piggybacked on Data or Null packets being transmitted.

For example, a Slot Allocation Request consists of the address of the mobile host requesting the allocation (6 bytes), the number of allocations requested (1 byte), a sequence number (2 bytes) and a timestamp (8 bytes). Therefore, the total size of an update message is 15 bytes.

As the Additional Info field is of fixed size, this implies that the number of updates currently ongoing using the atomic multicast protocol in TBMAC is bounded. Thus, the size of the Additional Info field impacts on the number of allocations and deallocations that can be ongoing in the cell at any one time. Obviously, this impacts on higher layers wishing to make such allocations and deallocations. Therefore, just as mobile hosts agree on the size and duration of the CFP and the CP, they must also agree on the size of this Additional Info field before the start of the TBMAC protocol.

The above paragraphs describe the header of a CFP slot transmission. The header used in a CP transmission is much simpler and consists of a source and destination address and a request field indicating that the mobile host is requesting one or more CFP slots and a sequence number.

3.3.3 Non-empty CFP

The functionality provided by TBMAC to manage the allocation of CFP slots in a non-empty cell can be broken into four parts: requesting a slot, allocating a slot, deallocating a slot, leaving a cell and inter-cell communication.

Requesting a slot

When a mobile host enters or powers on in a non-empty cell and requires a slot in the CFP to be allocated to it, the mobile host first needs to learn which CFP slots have been allocated and which CFP slots are available. The mobile host waits for the beginning of the next CFP and then listens for one full CFP to pass before requesting a slot in the following CP. The Slot Bitmap field in each occupied CFP slot indicates the current allocation of CFP slots in the cell. By receiving at least one message in the CFP correctly, the listening mobile host obtains from the slot bitmap the current allocation of CFP slots in the cell and the position of each of these allocated CFP slots. The mobile host then requests a CFP slot to be allocated to it by sending a message in the CP.

The approach taken by the TBMAC protocol is for a mobile host to request a slot by choosing a random slot in the CP and broadcasting to every mobile host requesting a slot to be allocated to it. By randomly choosing a slot in the CP, a degree of uncertainty has been introduced by the TBMAC protocol. If two or more mobile hosts choose the same slots in the CP, then their requests for a CFP slot will result in a collision.

However, the probability of two or more hosts not colliding in the CP can be calculated provided that a model of the arrival rate of mobile hosts into the cell is known. Varying the size of the CP results in a corresponding change in this probability. Increasing the size of the CP results in an increase in this probability while decreasing the size of the CP results in a corresponding decrease.

It is possible to reduce further the possibility of collisions in the CP by modifying

the way that hosts contend with each other. These extensions to the TBMAC protocol will be discussed in section 3.3.5.

Allocating a slot

Mobile hosts with CFP slots allocated and that correctly receive a request in the CP, propose an update to the allocation of CFP slots by inserting an atomic multicast related message into the Additional Info field of each transmission in its CFP slot(s).

The address in this atomic multicast message is set to the address of the mobile host making the request. The delivery time of this message corresponds to the time of the request in the CP slot plus the delivery delay of the agreement protocol (which in the TBMAC is $2 * (CFP + CP)$). The sequence number used in the atomic multicast message corresponds to the sequence number received from the CP request.

The reason for taking the sequence number and calculating the timestamp from the request in the CP is to ensure that if two mobile hosts with CFP slots allocated receive this slot request, they propose the same update to the allocation of CFP slots. This prevents a request for a slot resulting in the allocation of more than one CFP slot.

After the delivery of this atomic multicast message, the mobile hosts in the cell update their view of the allocation of CFP slots and the requesting mobile host can then access its allocated CFP slot. In the absence of collisions, the time bound before a mobile host can transmit in the CFP slot allocated to it equals $5 * (CFP + CP)$.

The above description of the set of steps that a mobile host must execute to be allocated a CFP slot does not take into consideration whether or not the mobile host should be allocated the CFP slot. In other words, all requests for CFP slots in the above description would be accepted and result in atomic multicasts being executed. To prevent such a situation from arising, the TBMAC protocol provides functionality to implement admission control on the requests for allocation of CFP slots.

The interface between a higher layer and the TBMAC protocol enabling this ad-

mission control feature will be covered in the next chapter. Briefly, it is possible for a higher layer above TBMAC to register a callback with the TBMAC protocol to be queried on whether or not to proceed with the allocation of a CFP slot. The execution of this callback function should finish its execution before the reception of the next CP request or before the receiving mobile host has to transmit the request in its allocated CFP slot. If such a call back has not been registered, then the TBMAC protocol proceeds with the allocation of the slot.

Deallocating a slot

The simplest way for a mobile host to deallocate one of its CFP slots is to atomically multicast a message requesting the deallocation. This could happen if a mobile host is powering down or realises, based on location information, that it is leaving the cell.

When deallocating such a CFP slot, the mobile host allocated the slot creates a Slot Deallocation update and inserts it into its Additional Info field during each of its CFP slot transmissions. The delivery time for this update corresponds to 2 TBMAC rounds after the first CFP slot that this mobile host transmits this update in. Instead of specifying the address of the mobile host that wishes to deallocate its slot, the position of the slot in the CFP is included as part of the Slot Deallocation update. This allows a listening mobile host to correctly deallocate the CFP slot without knowing the address of the mobile host that has been allocated this slot.

It is more difficult to deallocate a CFP slot of a mobile host that has failed without firstly atomically multicasting a request to deallocate the CFP slot. In addition, we would also like to avoid deallocating a CFP slot of a mobile host that does not want its slot deallocated.

Each mobile host, that has been allocated a CFP slot, monitors the packet header of each of the other allocated CFP slots for correct reception. If a mobile host does not correctly receive a number of transmissions, above a certain threshold, in a CFP

slot from a failed mobile host, then it includes this information in the Additional Info field of each of its CFP slots indicating the failure of the mobile host to use its slot.

After receiving a majority of such indications from mobile hosts in the cell, a mobile host atomically broadcasts a Slot Deallocation update for the CFP slot to be deallocated. After the delivery of the atomic broadcast update, the CFP slot of the failed mobile host is deallocated by each mobile host in the cell.

The majority value used in this calculation is the smaller of two possible values: the current number of mobile hosts with CFP slots allocated that are not suspected to have failed and the number of mobile hosts that are not suspected to have failed when the failure of the mobile host allocated this CFP slot is first noticed. Choosing this majority value prevents the possibility of a mobile host waiting for indications from mobile hosts that have recently arrived into the cell and were allocated a CFP slot and also from mobile hosts that have also failed or recently departed the cell.

The approach that TBMAC takes to detecting the failure of a mobile host to use its slot is similar to how the TTP protocol detects a failure of a TTP node. In both TTP and TBMAC, a majority of acknowledgments from the active hosts is needed before a failure is deemed to have occurred. As TTP is designed for a wired medium, two TDMA rounds is sufficient to detect the failure of a TTP node. In contrast, TBMAC allows the threshold for detection of a failed host to be specified since the possibility of transient errors in a wireless environment is greater. The default value for this threshold parameter to the TBMAC protocol is 3. Thus a mobile host with a CFP slot that has failed will be detected after 3 TBMAC rounds before execution of the above deallocation begins.

Note that in the above description, the failure of a mobile host to use and/or deallocate its CFP slot is assumed to be an exceptional occurrence rather than standard behaviour. Note also that the TBMAC protocol does not provide any functionality for a higher layer slot manager to intercede to prevent the deallocation from proceeding.

It is possible, however, for a higher layer to set the threshold value for incorrect receipt of transmissions above which a mobile host is deemed to have failed to use a particular slot.

Leaving a cell

Before a mobile host departs from an old cell and enters a new one, the mobile host inserts a Leave update into its Additional Info and this is atomically multicasted to the other mobile hosts in the old cell indicating that it is leaving the cell. This Leave update gives remaining mobile hosts early warning about the impending departure of the mobile host. In addition to this Leave update, the departing mobile host also atomically multicasts Slot Deallocation updates for each of its allocated CFP slots. When the leave message is delivered, the departing mobile host switches to the radio channel of the new cell and resets all information related to the allocation of CFP slots from the old cell.

Inter-Cell Communication

As the area occupied by the mobile hosts has been divided up into geographical cells and each cell has been allocated a particular radio channel to use, an obvious question is how does a mobile host communicate with mobile hosts in neighbouring cells.

A simple solution to this problem would be to statically allocate the same CFP slot of two adjoining cells for communication between mobile hosts in these cells. When a mobile host wishes to use the inter-cell CFP slot, it atomically broadcasts a request to be allocated the CFP slot. Once this request is delivered, the mobile host begins to use the slot.

There is a problem with this approach since transmitting mobile hosts in each cell could collide with each others transmission. This is illustrated in Figure 3.4 where S1 in cell 1 has requested access to use the same inter-cell slot as S2 in cell 2. When S1

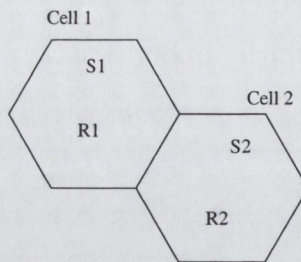


Fig. 3.4: Adjoining Cells

and S2 transmit, their packets collide.

If S1 uses cell 2's radio channel and S2 uses cell 1's radio channel then their transmissions will not collide. R1 will be listening for S2's transmission and R2 will be listening to receive S1's transmission. However, S1 will not be able to receive S2's transmission since S1 transmitted at the same time as S2.

The above problem can be avoided by using two CFP slots, one slot for communication from Cell 1 to Cell 2 and another slot from Cell 2 to Cell 1. As before, mobile hosts in each cell would atomically broadcast a request to be granted the inter-cell CFP slot. When a mobile host transmits in the inter-cell slot, all the mobile hosts in the adjoining cell will be listening for this transmission.

In a similar way to allocating a CFP slot, whether or not a mobile host should be allowed to transmit using one or more inter-cell slots is a decision that cannot be taken by the TBMAC protocol. The TBMAC protocol does provide a higher layer with information related to inter-cell slots, such as the number of inter-cell slots allocated, and functionality to give ownership of a particular inter-cell slot to a mobile host.

3.3.4 Empty CFP

In the previous section, mobile hosts with an allocated CFP slot in a non-empty cell can propose updates to the current allocation of CFP slots using the atomic multi-

cast protocol to transmit Slot Allocation and Slot Deallocation requests. An implicit assumption of the previous section was that the current allocation of CFP slots is consistent across all the mobile hosts with CFP slots allocated in the cell. If this assumption holds, then the Slot Allocation and Slot Deallocation requests transmitted using the atomic multicast protocol will update the current allocation of CFP slots correctly.

Therefore, two related questions are how do mobile hosts agree on an initial allocation of CFP slots and how is it ensured that this allocation is consistent? To answer these two questions, the problem of an empty cell in TBMAC needs to be addressed and how the first CFP slot is allocated in such a cell.

As mentioned in 3.1.3, TBMAC uses a bootstrapping protocol to allocate the first CFP slots in a cell. In the next sections, requirements for such a bootstrapping protocol will be covered, followed by a description, an example of the bootstrapping protocol in action and an analysis of the bootstrapping protocol used in TBMAC. The analysis of the bootstrapping protocol will give the underlying reasons why the protocol converges to a consistent allocation of CFP slots.

Bootstrapping Protocol Requirements

Potentially, an arbitrary number of hosts may enter an empty cell at approximately the same time and wish to execute the bootstrapping protocol. Thus, the bootstrapping protocol should allow a number of mobile hosts to reach an agreement on a consistent allocation of CFP slots. However, such a bootstrapping protocol should not preclude a lone mobile host in an empty cell from successfully allocating itself one or more CFP slots.

In addition, the bootstrapping protocol should attempt to address some of the problems associated with wireless communication covered in chapter 2 such as errors during transmission due to multi-path interference and collisions.

As TBMAC is concerned with providing time bounds on a mobile host gaining access to the wireless medium, the duration of the bootstrapping protocol should be bounded in time. Bounding the duration of the bootstrapping protocol and allowing an arbitrary number of mobile hosts to reach agreement on a consistent allocation of CFP slots is difficult to achieve with certainty. The approach the TBMAC bootstrapping protocol takes is to bound the duration of the protocol and to provide a corresponding probability of a consistent allocation of CFP slots to be agreed upon at the end of the bootstrapping period. In a similar way to providing a probability for mobile hosts colliding in the CP, a model of the arrival rate of mobile hosts into a cell is needed to calculate the probability of a consistent allocation of CFP slots being obtained.

Allocating a Slot

A simple approach would be for a mobile host to wait for another mobile host to transmit in the CP. These two mobile hosts could then negotiate with each other in the CP to agree on an allocation of slots in the CFP. The main problem with this approach is that two mobile hosts could choose the same CP slot in which to transmit and therefore corrupt each other's packets and therefore be unable to reach agreement. The likelihood of this occurring increases as the number of mobile hosts powering on within the cell increases and is also exacerbated by the fact that the CP is relatively small compared to the CFP. This approach also precludes a lone mobile host from allocating itself a CFP slot.

Instead of the above approach, after a mobile host listens for one full CFP period and discovers that it is in an empty cell, the mobile host switches to the bootstrapping protocol and generates a list of CFP slots to use as illustrated in Figure 3.5. Recall that each position in a mobile host's Slot Bitmap can be set to one of four possible values: Available, Owner, Other and Collision. For each CFP slot in the generated list, the mobile host sets the corresponding 2-bit value of its local Slot Bitmap to Owner.

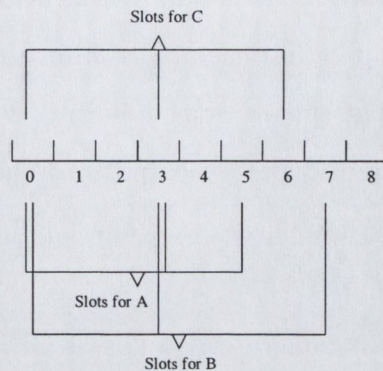


Fig. 3.5: Generated list of slots

For all other slots, the 2-bit value for that slot is set to Available.

In Figure 3.5, mobile host A's list consists of slots 0, 3 and 5 while mobile host B's list consists of slots 0, 3 and 7. Mobile host A would also set position 0, 3 and 5 of its local slot bitmap to Owner and set every other position to Available. By generating a list of slots to use, the possibility of all slots in two or more mobile hosts' lists colliding is reduced.

Converging to a Consistent Allocation of Slots

When two or more mobile hosts enter an empty cell and generate a list of CFP slots to use, each mobile host has an inconsistent view of the allocation of CFP slots. A simplified view of the bootstrapping protocol is that a mobile host executing the protocol assumes that none of the slots in its generated list are causing collisions until the mobile host is told by another mobile host that they are. On being told of one of its slots causing a collision, a mobile host refrains from using that slot. The procedure for a mobile host to detect that a slot is causing a collision is covered in the following paragraphs.

Note that following the above description of the bootstrapping protocol, a lone mobile host is able to bootstrap an empty cell by simply generating a random list of

CFP slots to use. Note also that an assumption of the following description of the bootstrapping protocol is that each mobile host executing the protocol has at least one slot in its generated list that is not conflicting with the slot of any other mobile host. Under certain constraints, the probability of each mobile host choosing at least one distinct slot and thus having at least one CFP slot at the end of the bootstrapping protocol can be calculated.

After a mobile host generates its list of CFP slots, the mobile host transmits in each CFP slot (including its local slot bitmap in each transmission) for a finite number of subsequent rounds of the TBMAC bootstrapping protocol. The mobile host also listens in each of the other CFP slots that it does not transmit in to gain knowledge about the presence of other mobile hosts and the CFP slots that they are using.

When a mobile host correctly receives a transmission from another mobile host, the receiving mobile host acquires knowledge about what CFP slots the transmitting mobile host is aware of from the slot bitmap in the transmitted packet. The receiving mobile host then updates its view of the allocation of CFP slots in the cell. The receiving mobile host then includes this new view of the allocation of CFP slots in each of its subsequent transmissions by including its local updated slot bitmap. The sequence of steps that a receiving mobile host uses to update its view of each slot in the CFP (i.e., each 2-bit in its local Slot Bitmap) and thus to converge on a view that is consistent with the other mobile hosts is described, using pseudo code, in figure 3.6.

It should be stressed that a mobile host that receives a transmission during the bootstrapping protocol checks each individual 2-bit value in the transmitted slot bitmap against its locally stored information about each individual CFP slot.

If a received 2-bit value is marked as Available in the received slot bitmap, then the receiving mobile host makes no changes to its corresponding local CFP slot (lines 2 & 3 of figure 3.6). Thus if the receiving mobile host is transmitting in this CFP slot, it will continue to do so. By marking the CFP slot as Available, the transmitting mobile


```

1: switch( rcvd_bit ):
2:     case Available:
3:         // Do Nothing
4:     case Owner:
5:         switch( local_bit ):
6:             case Available:
7:                 setCFPSlot( Other, position, src )
8:             case Owner:
9:                 setCFPSlot( Collision, position, NONE )
10:            case Other:
11:                checkCFPSlotAgainstAddInfo( src, position, pkt )
12:            case Collision:
13:                // Do Nothing
14:        case Other:
15:            switch( local_bit ):
16:                case Available:
17:                    owner = getOwnerFromAddInfoField( )
18:                    setCFPSlot( Other, position, owner )
19:                case Owner:
20:                    checkCFPSlotAgainstAddInfo( src, position, pkt )
21:                case Other:
22:                    checkCFPSlotAgainstAddInfo( src, position, pkt )
23:                case Collision:
24:                    // Do Nothing
25:            case Collision:
26:                setCFPSlot( Collision, position, NONE )

```

Fig. 3.6: Pseudo code for mobile host processing each 2-bit value in the slot bitmap of a bootstrap packet

host is indicating that it has yet to become aware of any mobile host transmitting in this slot.

If a received 2-bit value is marked as Owner in the received slot bitmap, the receiving mobile host checks the corresponding locally stored information about this slot (lines 5 to 13 in figure 3.6).

If the receiving mobile host has locally marked the slot as Available, then the receiving mobile host updates this information to reflect that the transmitting mobile host is transmitting in this slot. The receiving mobile host also stores the address of the transmitter in its local information about this slot (the Slot Owners array). When a receiving mobile host marks a position in its Slot Bitmap as Other, the receiving mobile host includes information, in the Additional Info field of each of its transmissions, indicating the mobile host that is transmitting in this slot. This information is only included in the Additional Info field of each packet until the end of the bootstrapping period.

In contrast, if the receiving mobile host has marked the local slot as Owner, then the transmitting and receiving mobile hosts are colliding in this slot. The receiving mobile host marks its local slot as causing a Collision and stops transmitting in this CFP slot in subsequent rounds of the bootstrapping protocol. If all the receiving mobile hosts CFP slots are marked as causing a Collision, then the receiving mobile host continues to listen for transmissions in each CFP slot and updates its local CFP slot information and waits for the bootstrapping protocol to end.

In a similar approach, if the receiving mobile host has marked its local CFP slot as Other, then the receiving mobile host checks the address of the transmitter of the packet against the locally stored address (line 11 in figure 3.6). If the two addresses conflict, the receiving mobile host marks the corresponding slot as causing a Collision. Otherwise, the receiving mobile host does not change its locally stored information about this slot.

Lines 14 to 24 in figure 3.6 deal with the case of how a receiving mobile host processes each 2-bit value in a transmitted slot bitmap marked as Other. If the corresponding 2-bit in the locally stored slot bitmap is marked as Available, then the receiving mobile host retrieves the owner of this slot from the Additional Info field of the transmitted packet and stores this information locally.

If the 2-bit value is locally marked as Owner, then the receiving mobile host retrieves the address from the Additional Info field of the packet and compares it against its own address. If the two addresses do not match, the receiving mobile host marks the corresponding value in its local slot bitmap as causing a Collision. Again, the receiving mobile host refrains from transmitting in this slot in subsequent rounds of the bootstrapping period.

If the 2-bit value is locally marked as Other, then the receiving mobile host checks the address in the Additional Info field against the locally stored address for this slot. If the two address do not match, then the receiving mobile host marks this slot as causing a Collision.

Finally, if a received 2-bit value is marked as Collision in the received slot bitmap, the receiving mobile host marks its corresponding local CFP slot as causing a collision (lines 25 & 26 of figure 3.6). If the receiving mobile host was transmitting in such a Collision slot, then the receiving mobile host refrains from transmitting in this slot.

Each mobile host executes the above steps on each 2-bit value of each correctly received CFP transmission during the bootstrapping protocol. At first glance, this would appear to place a heavy processing burden on each mobile host receiving each CFP slot transmission. However, on average over $3/8$ of this work load is trivial (lines 2 & 3, lines 12 & 13 and lines 23 & 24 of figure 3.6). Similarly, $3/8$ of this work load is relatively simple ((lines 6 & 7, lines 8 & 9 and lines 25 & 26 of figure 3.6).

Collisions in the above bootstrapping protocol are detected by a mobile host learning of a conflict between either itself and another mobile host or between two other

mobile hosts through the reception of packets and not by monitoring signal to noise ratio during each CFP slot. Using the latter approach would require the TBMAC protocol to be able to differentiate between a packet corrupted due to multi-path interference and a collision between two transmitting mobile hosts. If such an approach were followed, this could result in a mobile host erroneously marking a CFP slot as causing a collision when it is not.

Similarly, the problem of capture can be reduced using the collision detection algorithm of the bootstrapping protocol since mobile hosts defer from using any allocated CFP slots once they receive an indication through an intermediary mobile host that their CFP slot is causing a collision.

The bootstrapping protocol lasts for a finite number of rounds of the TBMAC protocol, M say. This allows a number of transmissions from a mobile host in a distinct CFP slot to be corrupted due to interference while still allowing this mobile host to continue using this distinct CFP slot in later rounds of the bootstrapping protocol.

If another mobile host arrives during these M rounds, then the mobile host must wait until the end of the M rounds, performing any necessary updates to its Slot Bitmap, before requesting a CFP slot to be allocated to it in the CP following the bootstrapping period. Similarly, if all CFP slots in a mobile hosts list are causing a collision, the mobile host must also listen until the end of the M TBMAC rounds, before requesting a CFP slot to be allocated to it in the next CP.

To allow an arriving mobile host to know the number of rounds of the algorithm left to be executed, it is necessary for the mobile hosts to include the current round of the bootstrapping period in the Additional Info field of the CFP packet transmission.

Convergence algorithm example

The above convergence algorithm is best illustrated with an example. In figure 3.7, mobile hosts A, B and C use the generated list of slots corresponding to figure 3.5 and

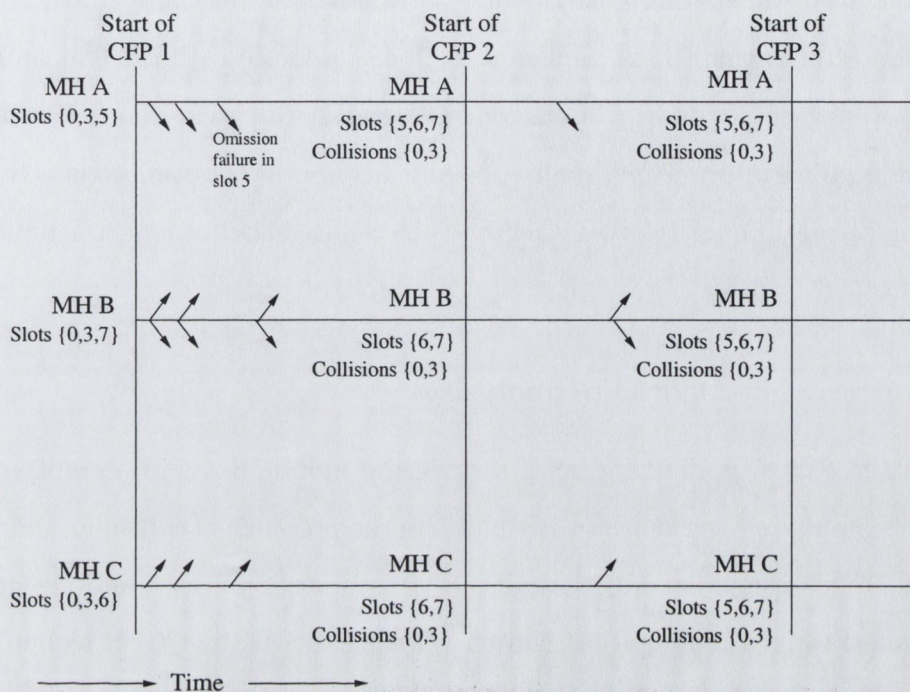


Fig. 3.7: Convergence of Slot Allocations

begin transmitting in each of their list of slots.

During the first CFP of the bootstrapping period, the transmissions in slot 0 and slot 3 of each of the hosts corrupt each other. The transmission of mobile host C in slot 6 and mobile host B in slot 7 are successful while a transient error causes the transmission of mobile host A in slot 5 to be corrupted. Mobile hosts A and C update their view of the allocation of slots (and of collisions) upon receiving the transmission from B. Similarly, A and B update their view when they receive C's transmission.

At the end of this CFP (i.e. the beginning of the second CFP), mobile hosts B and C have a mutually consistent view of the allocation of each others CFP slots in the cell. This view is inconsistent with mobile host A since B and C are unaware of the presence of mobile host A. However, all the hosts do have a correct view of the CFP slots that are causing collisions.

During the second CFP, a correct transmission by mobile host A would result in B and C converging to A's view of the allocation of slots. If A's transmission was received incorrectly at C due to interference, C could still learn about this allocation through B including information about this allocation in its transmissions (through bit 5 of B's slot bitmap being set to Other and A's address being transmitted in the Additional Info field by B).

Convergence algorithm analysis

At the end of bootstrapping protocol, the mobile hosts in a empty cell converge on a consistent view of their Slot Bitmaps (in the presence of collisions and packet corruption due to transient errors). Again, this assumes that each mobile host executing the bootstrapping protocol has chosen at least one distinct CFP slot in its generated list of slots. The underlying reason the above algorithm converges on a consistent view of the Slot Bitmap is because it is an example of an epidemic algorithm [23].

The Slot Bitmap of a mobile host, transmitted in a CFP slot, represents all the updates that that mobile host has performed on its allocation of CFP slots. In a similar way to [23], the probability of one or more mobile hosts having an inconsistent view of the allocation of CFP slots can be calculated. Therefore, by calculating the probability of a collection of mobile hosts each choosing a distinct slot and then calculating the probability of these collection of mobile hosts converging on a consistent view of CFP slots, the probability of the time bounds, from section 3.1.3, being obtained can be calculated (see section 3.4).

The goal of the convergence algorithm is to detect collisions when they occur and to subsequently attempt to avoid these collisions recurring. Thus the convergence algorithm is similar to the contention resolution schemes discussed in section 2.2.2 of chapter 2. The main difference between the convergence algorithm and the contention resolution schemes is that a mobile host can learn of the existence of another mobile

host using a particular CFP slot or of a collision in a CFP slot from an intermediary. In the convergence algorithm example in section 3.3.4, mobile host C could possibly learn about the presence of mobile host A from the intermediary B without mobile host C ever receiving a transmission from mobile host A.

In the convergence algorithm, there is a possibility (however small) of the contention not being resolved. This occurs when all the CFP slots of every mobile host are conflicting. In this instance, the mobile hosts will continue using their slots unaware that their slots are causing a number of collisions. These collisions will continue throughout the bootstrapping period and afterwards. A new mobile host enters the cell, at any stage during this time, will detect an empty cell and begin the bootstrapping protocol. If the colliding mobile hosts receive a transmission from this new mobile host, then they relinquish their allocated slots and wait for the new mobile host to finish executing the bootstrapping protocol.

Deallocating a Collision Slot

At the end of the bootstrapping period, slots marked as causing a collision are garbage collected and converted into available slots that can be allocated to requesting mobile hosts. TBMAC takes an approach similar to deallocating a CFP slot described earlier. A colliding slot is deallocated by one or more mobile hosts executing the atomic multicast protocol.

3.3.5 Extensions

It is possible to further reduce the possibility of collisions in the CP. One way to reduce this possibility of collisions would be to further subdivide the cell as illustrated in Figure 3.8. Each subdivision of the cell would correspond to one or more slots in the CP. When a mobile host powers on, it calculates which cell it is in and then which subdivision it is in. It then sends a request in the CP slot corresponding to the subdivision that it

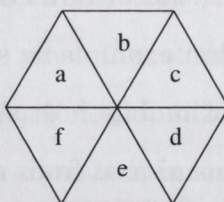


Fig. 3.8: Subdivision of a Cell

is in. By increasing the number of subdivisions, we decrease the possibility of two or more mobile hosts powering on in the same subdivision and therefore further reducing the possibility of collisions in the CP.

A second approach to reducing these collisions would be to use some of the CSMA techniques described in section 2.2. Before transmitting in the CP slot, mobile hosts would sense the wireless medium for a random period of time. If the mobile host senses the wireless medium is busy, then the mobile host would defer its transmission, otherwise it would transmit.

Another possible extension to the TBMAC protocol would be to allow a dynamic number of CFP slots to be allocated for inter-cell communication between two adjoining cells. There are two possible approaches to enabling this.

Firstly, a higher layer, possibly the routing layer, could coordinate the allocation of such a slot in both cells. This would then result in two atomic broadcasts being executed, one in each adjoining cell, to allocate an overlapping CFP slot to enable the communication between the adjoining cells to take place.

A second approach involves adapting the existing atomic broadcast protocol to work across the boundary between the two adjoining cells. The delay to delivery, Δ , of the atomic multicast protocol needs to be increased to reflect the fact that agreement is required across two cells as opposed to within just one cell. However, the atomic multicast protocol [20] has been designed originally to work in both a single hop scenario

(within a cell) and a multi-hop scenario (across multiple cells).

This second approach has the advantage of not requiring a higher layer to coordinate the allocation of the inter-cell slot. The disadvantage of using this approach is that the adapted atomic broadcast does not take into account whether mobile hosts are present or not in the adjoining cell. This could result in an inconsistent view of the number of inter-cell slots arising in each neighbouring cell.

The particular inter-cell slot allocated needs to be the same in both cells, otherwise transmissions in this inter-cell slot could result in collisions in the adjoining cell. The issue of whether such a slot should be allocated or not is again an issue for a higher slot management layer above TBMAC.

3.4 Protocol Inaccessibility

In a computer system, inaccessibility [67] is when certain components are temporarily unresponsive without having actually failed. Thus, an inaccessibility study of a computer system analyses the periods of time when the computer system is unable to provide its designated service.

A classic example of inaccessibility in wired networking is token loss and recovery in a token ring network [68]. In a token ring network, the largest inaccessibility period occurs when the token is lost, the hosts connected to the token ring network must detect when this occurs and then instigate a sequence of steps to recover the token. Hosts connected to the token ring are unable to transmit packets during this inaccessibility period.

In TBMAC, an important aspect of inaccessibility occurs when a mobile host powers on in a cell or moves from one cell to another. There are two distinct inaccessibility periods depending on whether the cell that the mobile host has just entered or powered on in is empty or not. In the non-empty cell case, a mobile host is unable to transmit

during the time period between when a mobile host requests a slot and this request results in the allocation of a CFP slot. In the empty cell case, a mobile host is unable to transmit during the bootstrapping period.

During previous sections in this chapter (3.3.3 and 3.3.4), the time bounds for these inaccessibility periods were introduced. Obviously, since the number of mobile hosts entering the cell is unknown a priori, this time bound is not certain (probability equal to 1) and therefore we derive the probability of each time bound being obtained. Before deriving these probabilities, we discuss how arrivals and departures into a cell are modeled.

3.4.1 Modeling Arrival/Departures

Arrival and Departure rates of mobile hosts into a TBMAC cell will depend on the movement patterns of the mobile hosts using the TBMAC protocol. For the purposes of the following inaccessibility analysis, we model the arrival rate of mobile hosts into the cell using a Poisson process [78]. A similar analysis could be carried out if a different model of the arrival rate into a cell is known. Let λ be the arrival rate of mobile hosts into the cell, then the probability of k mobile hosts being in the cell in the time interval $[0, t]$ is given by

$$Prob(k \text{ hosts present}) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad (3.1)$$

The time bound for an arriving mobile host acquiring a CFP slot depends on whether the cell, that the mobile host is entering, is empty or not. We will first consider the case of the cell being non-empty and then address the case when the cell is empty.

3.4.2 Non empty cell

As discussed in section 3.3.3, the time bound, in the absence of collisions, occurs when a mobile host enters a non-empty cell just as the CFP begins. The time bound before a mobile host has a CFP slot allocated to it equals $4 * (CFP + CP)$.

By choosing a random slot in the CP, two or more mobile hosts could choose the same slot and therefore collide. We would now like to derive the probability of avoiding these collisions.

Let the number of slots in the CP be N_{CP} , then a mobile host randomly chooses a slot with probability $\frac{1}{N_{CP}}$. Given k mobile hosts arriving in the cell, the probability of these k mobile hosts choosing different slots equals

$$Prob(k \text{ distinct slots chosen}) = \frac{(N_{CP})(N_{CP}-1) \cdots (N_{CP}-(k-1))}{N_{CP}^k} = \frac{N_{CP}!}{(N_{CP}-k)!N_{CP}^k} \quad (2)$$

Therefore, combining probabilities (1) and (2) we get,

$$Prob(k \text{ hosts in cell and no collisions occur}) = Prob(k \text{ hosts present}) * Prob(k \text{ distinct slots chosen})$$

Finally in this section, the probability that the time bound is obtained is equivalent to there being no collisions in the cell during the CP and is given by

$$Prob(\text{time bound is obtained}) = \sum_{k=1}^{\infty} Prob(k \text{ hosts in cell and no collisions occur}) \quad (3)$$

3.4.3 Empty cell

Again, when a mobile host enters an empty cell, the time bound, in the absence of collisions, occurs just as the CFP begins. The mobile host must then wait until the end of this CFP and the following CP before listening at the beginning of the next CFP. On realising that no mobile hosts have been allocated slots in the CFP, the mobile host

generates a list of random CFP slots and executes the bootstrapping protocol described in section 3.3.4 for a number of CFPs.

The time bound before a mobile host has a CFP slot allocated to it equals:

$$2 * (CFP + CP) \quad (4).$$

However, the mobile host will not have a consistent view of the allocation of CFP slots until M rounds of the bootstrapping protocol have passed. Therefore, the more important time bound is:

$$(M + 2) * (CFP + CP) \quad (5).$$

Now the probability of a mobile host having a consistent view on the allocation of CFP slots at the end of time bound (5) depends, firstly, on the mobile host choosing at least one CFP slot that does not collide with any other CFP slot chosen by any other mobile host. Secondly, the probability also depends on the probability of the contending mobile hosts converging on a consistent view of the allocation of CFP slots at the end of the bootstrapping period.

We would now like to derive the probability of time bound (4) being obtained. Let N_{CFP} be the number of slots in the CFP and let j be the number of slots that each of the k mobile hosts generate at the start of the bootstrapping period. Then, the probability of at least one of the j generated slots of each of the k mobile hosts not colliding with any other slot of the other mobile hosts is

$$\begin{aligned} & \text{Prob(All } k \text{ hosts generate at least 1 distinct slot)} \\ &= \left(\frac{(N_{CFP}) \cdots (N_{CFP} - (k-1))}{N_{CFP}^k} \right) \left(\left(\frac{(N_{CFP} - k)}{(N_{CFP} - 1)} \right)^k \left(\frac{(N_{CFP} - k - 1)}{(N_{CFP} - 2)} \right)^k \cdots \left(\frac{(N_{CFP} - k - ((j-1) - 1))}{(N_{CFP} - 1 - ((j-1) - 1))} \right)^k \right) \quad (6) \\ &= \left(\frac{N_{CFP}!}{(N_{CFP} - k)! N_{CFP}^k} \right) \left(\frac{(N_{CFP} - k)! (N_{CFP} - 1 - (j-1))!}{(N_{CFP} - 1)! (N_{CFP} - k - (j-1))!} \right)^k \quad (7) \end{aligned}$$

The first term of the above formula (in both (6) and (7)) represents the probability of k mobile hosts each choosing 1 distinct slot. Note that the first term of the above

formula corresponds to the probability derived in (2) replacing N_{CP} in formula (2) with N_{CFP} .

Once these k collision free slots have been allocated out of the available N_{CFP} slots, there are $N_{CFP} - k$ CFP slots remaining. The second term represents the probability of the k mobile hosts choosing their remaining $(j - 1)$ slots in these $N_{CFP} - k$ slots and thus not collide with the first k slots. In (6), after each of the k mobile hosts have chosen their first slot to be collision free, each of the k mobile hosts must choose their second slot in one of $N_{CFP} - k$ possible slots and they choose with probability $\frac{1}{N_{CFP}-1}$. After choosing their second slot in the $N_{CFP} - k$ possible slots, the k mobile hosts must then choose their third slot in one of $N_{CFP} - k - 1$ possible slots and they each choose this third slot with probability $\frac{1}{N_{CFP}-2}$. For each mobile host to have chosen at least 1 distinct slot, the k mobile hosts must choose their remaining $j - 3$ slots in the $N_{CFP} - k - 2$ possible slots.

Similar to before, the probability of k mobile hosts being in an empty cell at time t and each of these k hosts generating at least one collision free slot is

$$\begin{aligned} & \text{Prob}(k \text{ hosts present and each has a distinct slot}) = \\ & \text{Prob}(k \text{ hosts present at time } t) * \text{Prob}(k \text{ hosts generate 1 distinct slot}) \end{aligned}$$

The probability of time bound (4) being obtained in an empty cell is

$$\begin{aligned} & \text{Prob}(\text{time bound (4) being obtained}) = \\ & \sum_{k=1}^{\infty} \text{Prob}(k \text{ hosts in cell and each has a distinct slot}) \quad (7) \end{aligned}$$

In addition to the probability of time bound (4) being obtained, to calculate the probability of time bound (5) being obtained, we also need to calculate the probability of all the mobile hosts having a consistent view of the allocation of CFP slots after executing the collision detection steps for the M rounds of the TBMAC protocol corresponding to the bootstrapping period.

A mobile host remains unaware of an update to the allocation of CFP slots after the $i + 1^{st}$ round of the bootstrapping protocol if it was unaware of the update in the i^{th} round and any transmissions by mobile hosts that are aware of the update are corrupted during the $i + 1^{st}$ round.

Let P_{pkt} be the probability of a packet being corrupted. As in [23], let p_i be the probability of a mobile host, after the i^{th} bootstrap round, remaining unaware of an update to the allocation of slots. Then, $p_1 = P_{pkt}$ and the probability, p_{i+1} , the mobile host remains unaware of this update after the $i + 1^{st}$ TBMAC round is

$$p_{i+1} = p_i (P_{pkt})^{k(1-p_i)} \quad (8)$$

where k , from before, is the number of mobile hosts in the cell. The final term in formula (8) corresponds to the probability that the transmissions of mobile hosts, that know about the update, are all corrupted in the $i + 1^{st}$ TBMAC round of the bootstrapping period. The exponent of this final term, $k(1 - p_i)$, corresponds to the number of mobile hosts that are aware of the update after the i^{th} round.

After M rounds of the bootstrapping protocol, p_M is the probability of a mobile host being unaware of an update. Thus, $1 - p_M$ is the probability of a mobile host having a consistent view of the allocation of CFP slots after M TBMAC rounds. Note that p_i converges rapidly to 0 when P_{pkt} is small (less than 0.1) and i increases.

This formula differs from the push anti-entropy algorithm in [23] due to the fact that communication is point-to-point in [23] while every mobile host in the TBMAC protocol processes the CFP header of a correctly received packet.

3.5 Summary

This chapter gave a high level overview of the TBMAC protocol design. TBMAC is principally concerned with giving mobile hosts time bounded access to the wireless

medium and this chapter highlighted the steps taken by mobile hosts using the TBMAC protocol to be given predictable access to the wireless medium. The time bounds (and their associated probabilities of being obtained) for a mobile host to obtain predictable access to the medium were derived. Some of the issues, such as admission control, that the TBMAC protocol does not explicitly deal with were also highlighted. However, the TBMAC protocol does allow a higher layer to intercede during the operation of the protocol to tailor admission control to their specific needs. The interface provided by the TBMAC protocol to allow intercession by higher layers will be discussed in the next chapter.

Chapter 4

Implementation

The previous chapter gave an overview of the design of the TBMAC protocol. This chapter presents a realisation of this design implemented in the NS2 network simulator [1],[26]. Firstly, a brief overview of the NS2 simulator is given in section 4.1. Following this, the procedure to initialise the TBMAC protocol is described in section 4.2. In section 4.3, a detailed description is given of the functionality provided by TBMAC to manage the allocation and deallocation of CFP slots, including the data structures used. The chapter concludes with a description of how communication is achieved in TBMAC followed by a summary.

4.1 Implementation Environment

The NS2 network simulator is an object oriented discrete event simulator developed as part of the DARPA funded Virtual InterNetwork Testbed (VINT) project. VINT is a collaborative project involving USC/ISI, Xerox PARC, LBNL, and UC Berkeley.

NS2 supports a variety of both wired and wireless protocols at different levels in the protocol stack. A protocol executes in the simulator by scheduling discrete events related to the given protocol to occur at a simulated virtual time. Some examples


```

        int get_cfp_slot_size( )
    err set_cfp_slot_size( int size )
        int get_cp_slot_size( )
    err set_cp_slot_size( int size )
        int get_cfp_pkt_duration( )
    err set_cfp_pkt_duration( double duration )
        int get_cp_pkt_duration( )
    err set_cp_pkt_duration( double duration )

```

Fig. 4.1: Query/Change methods

of such events are the transmission of a packet at a mobile host and the corresponding reception of this packet at another mobile host within transmission range after a duration of virtual time related to the propagation delay.

The NS2 simulator supports two different programming languages, C++ [77] and OTcl [3] (which is an object-oriented extension of the Tcl scripting language [56]). Typically, C++ is used to develop a particular protocol and OTcl is used for rapid prototyping using the protocols already implemented in the simulator. The TBMAC protocol is implemented in C++ and the following sections will describe this implementation.

4.2 Initialisation

Before the TBMAC protocol starts, the participants in the protocol are required to agree on a number of pieces of information. Firstly, the participants should agree on the number of slots in the CFP and in the CP. In addition to the number of slots, participants should also agree on the duration (size of packet transmitted) of each CFP and CP slot. The TBMAC implementation in NS2 provides a number of methods to both query and change the underlying size and duration of a TBMAC round as shown in table 4.1.

Secondly, the participants also need to divide the geographical area that they cover into a set of geographical cells. As discussed in the previous chapter, the TBMAC protocol does not maintain this geographical information as it is maintained at a layer above TBMAC. A set of numbered radio channels (CDMA spreading codes) are then allocated to these geographical cells to ensure spatial reuse. To ensure correct functioning of communication using the TBMAC protocol within and between cells, TBMAC needs to support at least this number of radio channels.

In addition to these radio channels, TBMAC protocol participants must also agree on the static inter-cell slots that will be used between each cell and its neighbouring cells.

```
err change_in_geocell( int new_radio_channel,
                      inter_cell_slot list[ ],
                      int list_length )
```

Fig. 4.2: Method to indicate a change in layout of neighbouring cells

Before the TBMAC protocol begins executing, a higher layer above TBMAC, using the above method, informs the TBMAC protocol running on the mobile host of the current cell that the host is in. The current cell is equivalent to the `new_radio_channel` parameter to this method. A list of the neighbouring cells of this current cell and the corresponding static inter-cell slots to allow communication between each of these neighbouring cells and the current cell are also passed to this method using the second parameter. When a mobile host moves from one cell to another, the TBMAC protocol is informed of the change of cell (and a corresponding change in the inter-cell slots to use) using this same method.

Finally, a slot manager can register with the TBMAC protocol to be notified of when the allocation of CFP slots is changed. Registering a slot manager is achieved using the `set_tbmgr_mgr( )` method below. The `get_tbmgr_mgr( )` method retrieves


```
void set_tbmgr( SlotMgt * arg ) )
SlotMgt * get_tbmgr( ) )
```

Fig. 4.3: Initialise/Retrieve the slot manager of the TBMAC protocol

any already registered slot manager.

As the TBMAC protocol is written in the NS2 simulator, the SlotMgt parameter to the method `set_tbmgr( )` (and returned by the `get_tbmgr( )` method) is a virtual base class that slot manager instances inherit from. The SlotMgt base class represents the interface used by the TBMAC protocol to inform a higher slot management layer of important events in the operation of the TBMAC protocol. One such event is when a request for a CFP slot is received, other events include a change in the allocation of CFP slots.

4.3 Slot management

This section covers the main data structures used by the TBMAC protocol to allow the management of CFP slots and also includes a discussion of the interaction between the TBMAC protocol and a higher slot management layer when slots are allocated and deallocated.

4.3.1 Data structures

The most important data structure, illustrated in figure 4.4, in the TBMAC protocol is used to represent the state of an allocated CFP slot. An array of such data structures is then used to represent the state of all the slots in the CFP. This array is replicated at each mobile host in the cell.

The state variable in the CFP_Slot_Info structure represents the four possible states that the particular CFP slot can be in. From chapter 3, the four possible states are

CFP_Slot_Info
state owner slot_ref inter_cell is_static channel_num pkt_rcvd_correctly lost_pkt_count ack_count ack_list

Fig. 4.4: Information related to a CFP slot

Available, Owner, Other and Collision. The owner variable stores the address of the mobile host that is currently allocated that particular CFP slot.

As an instance of the CFP_Slot_Info structure represents an allocated CFP slot, the TBMAC protocol passes the slot_ref variable to higher layers as a reference to the CFP_Slot_Info structure. This allows a higher layer to query and/or manipulate the state of a particular CFP_Slot_Info structure using this reference.

Following the slot_ref variable in the CFP_Slot_Info structure is a boolean variable, inter_cell, that indicates whether the allocated CFP slot is for inter-cell communication (inter_cell equals true) or not. If the allocated slot is used for inter-cell communication, then the slot can be either statically or dynamically allocated, which is indicated by the is_static variable (true or false respectively). To allow inter-cell communication to occur, the TBMAC protocol also stores the number of the radio channel to switch to when a host wishes to transmit in this inter-cell slot.

Finally, the CFP_Slot_Info structure maintains information to detect if a CFP slot

is being used for transmission. This failure detection algorithm was described in section 3.3.3. The `pkt_rcvd_correctly` variable indicates whether the previous transmission in the corresponding CFP slot was received correctly or not. If not, then the `lost_pkt_count` variable is used to count the number of TBMAC rounds that have passed since a transmission was successfully received in this slot. When this `lost_pkt_count` variable exceeds a configurable threshold value, the mobile host adds its address to the `ack_list` and increases the `ack_count` by one. An indication of the possibility of a failure is then included in each subsequent allocated CFP slot. When a mobile host receives a similar indication from another mobile host, the receiving mobile host adds the address of the transmitting host to the `ack_list` and increments the `ack_count` variable by one. When a mobile host receives indications from a majority of mobile hosts that have CFP slots allocated to them, then the mobile host initiates the deallocation of the failed CFP slot.

In addition to the array of `CFP_Slot_Info` structures, the realisation of the TBMAC protocol provides an internal implementation of Flaviu Cristian's atomic multicast protocol [20]. The most important data structure in the atomic multicast protocol is the hash table that is used to store the messages, sent using the protocol, to be delivered when the delivery time of the message arrives.

The implementation of this hash table data structure is modified to reflect its use in the TBMAC protocol. In TBMAC, protocol messages can only be transmitted in slots in either the CFP or the CP. Therefore, TBMAC protocol messages sent using the multicast protocol are inserted into one of $(CFP + CP)$ possible positions in the hash table.

At each of these positions, a number of protocol messages can be stored in the order of their delivery time. Similarly to Cristian's atomic multicast protocol, if two messages have the same delivery time, then the address of the host that originally transmitted each protocol message is used to order the messages. In the event of two

messages having the same delivery time and the same originator, the sequence number of the item being inserted is then used to provide a unique order.

4.3.2 Querying the protocol

As the TBMAC protocol executes, CFP slots will be allocated and deallocated. Layers above the TBMAC protocol performing admission control and ensuring quality of service will want to query the TBMAC protocol for the number of slots allocated and also to which mobile hosts these slots have been allocated.

```
err get_allocated_slots( int * no_slot_refs,  
                        slot_ref2addr list[ ],  
                        int list_length )
```

Fig. 4.5: Retrieve a list of references to allocated slots

Using the above method, a higher layer will obtain the number of slots allocated from the first parameter to the method. The second parameter returns a mapping from the references of allocated CFP slots to the mobile hosts that have been allocated these slots. The third parameter is used to ensure that the second parameter has sufficient space to store the mapping for all the allocated CFP slots.

Similarly, higher layers will also need to query the TBMAC protocol for the number of inter-cell slots that have been allocated, which mobile hosts have been allocated these inter-cell slots and which adjoining cell each inter-cell slot will communicate with. A similar method to the previously mentioned `get_allocated_slots( )` method is used for this task. The only difference is that a mapping is returned from a reference of an allocated inter-cell slot to both the mobile host that is using the slot and the neighbouring cell number that the inter-cell slot is being used to communicate with. As before, the third parameter is used to ensure that the second parameter has sufficient space.


```
err get_inter_cell_slots( int * no_slots,
                        slot_ref2cell_and_addr list[ ],
                        out list_length )
```

Fig. 4.6: Retrieve a list of references to allocated inter-cell slots

The last issue in this section concerns how higher layers become aware of a change in the allocation of CFP slots. One option would be for the higher layer to poll the TBMAC layer for any changes in the allocation of slots using the two methods just introduced.

Another option would be for the higher layer to request the TBMAC protocol to inform it when a change to the allocation of CFP slots is occurring. The different type of changes, covered in chapter 3, to the allocation of CFP slots that can occur include allocation/deallocation of a CFP slot and allocation/deallocation of an inter-cell CFP slot. The interaction between the TBMAC protocol and a higher layer for each of these four possible scenarios will be discussed in the following sections.

4.3.3 Allocating a slot

The sequence of steps to allocate a CFP slot and the interaction between the TBMAC protocol, a slot manager and the hash table of the atomic multicast protocol is illustrated in figure 4.9.

A mobile host that enters a cell requests the TBMAC protocol to allocate a CFP slot to it. This request could be generated by any layer in the protocol stack but would typically be sent by a slot manager running on the mobile host. This request to the TBMAC protocol is done using the method:

A reference to the CFP slot to be allocated is returned to the caller in the first parameter to this method. The second parameter to this method specifies that a request for a CFP slot is being made (as opposed to an inter-cell slot). The third


```
err req_cfp_slot( int * slot_ref, int req_type, int address
                int additional_info )
```

Fig. 4.7: Request the TBMAC protocol to allocate a slot

parameter is the address of the mobile host that will be allocated the slot. The last parameter to this method is used when a request for an inter-cell slot is made (see section 4.3.5).

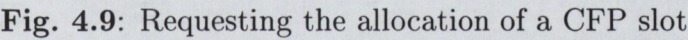
On receiving this request, the TBMAC protocol waits for the next CP, chooses a slot in the CP and transmits a protocol message requesting to be allocated a CFP slot. The TBMAC protocol includes in this request the address of the mobile host making the request, the delivery time of the request, the type of the request and a sequence number to provide a unique ordering. The delivery time of the request corresponds to the CP slot, which the request was transmitted in, a number of TBMAC rounds in the future.

On reception of this request, the TBMAC protocol, running on a mobile host that has a CFP slot allocated, queries the slot manager on whether to proceed with the request or not by invoking the following method of the slot manager. If no such slot manager has registered with the TBMAC protocol, then the TBMAC protocol begins the allocation of the CFP slot.

```
boolean mgr_notify_cfp_request( int slot_ref, int address,
                              double delivery_time )
```

Fig. 4.8: Retrieve a list of references to allocated slots

A slot manager can at this point choose to allow the request for a CFP to proceed by returning true. Therefore, the slot manager decides on the admission of mobile hosts requesting CFP slots. The slot manager could also take into account quality of



When the TBMAC protocol is allowed to proceed with a received request, the protocol then inserts the request for a slot into the hash table of the atomic multicast protocol and piggybacks this request on each of its subsequent transmissions in its CFP slots until the delivery time of the request arrives.

103

has already deemed it acceptable to proceed with the request. Thus, the TBMAC protocol assumes that this slot manager has taken into account any quality of service considerations before allowing the allocation to proceed.

Another reason to not notify the slot manager is due to the nature of the atomic multicast protocol being executed. As the atomic multicast protocol is a flooding protocol, either the TBMAC would inform the slot manager every time it receives a duplicate of this request or the TBMAC protocol maintains state information about requests that it has informed the slot manager about. In the first case, the slot manager must deal with the extra complexity of detecting duplicates. In the second case, the complexity of the TBMAC protocol is increased by moving the detection of these duplicates into the TBMAC protocol.

During the execution of the TBMAC protocol, the hash table of the atomic multicast protocol is queried by the TBMAC protocol for any items in the hash table whose delivery time has arrived. As highlighted in section 4.3.1, the delivery time of an item in the hash table corresponds to one of $(CFP + CP)$ possible positions. Therefore, it is only necessary for the TBMAC protocol to check that the delivery time of an item has arrived once in each slot in a round of the TBMAC protocol.

As illustrated in figure 4.10, when the delivery time of a request arrives, the TBMAC protocol removes the corresponding item from the atomic multicast hash table and then updates its allocation of CFP slots. After completing this update, the TBMAC protocol informs the slot manager, if one has been registered, of the update using the method below. This is another method of the virtual base class SlotMgt mentioned previously.

Here, the `slot_ref` parameter corresponds to the previous `slot_ref` passed to the slot manager by either the `req_cfp_slot( )` or the `mgr_notify_cfp_request( )`.

As discussed in section 3.3.4 of chapter 3, it is possible for a request for a CFP slot in the CP to collide with another request. This collision would then prevent a mobile host with a CFP slot from receiving the request correctly. This would in turn

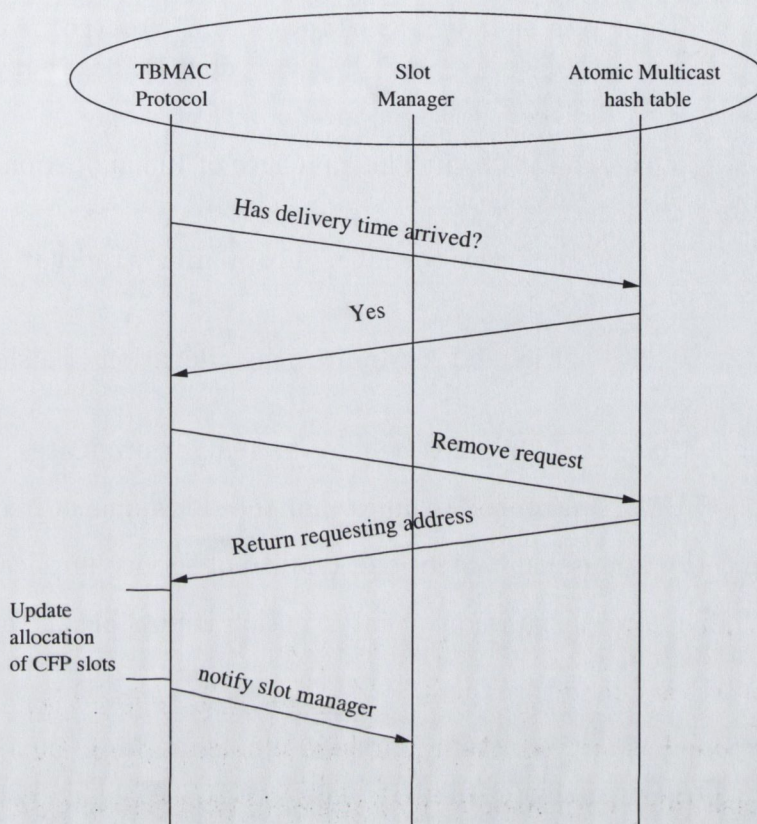


Fig. 4.10: Delivering the allocation request for a CFP slot

prevent the allocation of a CFP slot to the requesting mobile host from proceeding and the corresponding notification of a successful allocation. A requesting mobile host can monitor the progress of its request by calling the method

The `slot_ref` parameter passed to this method must correspond to the `slot_ref` parameter passed back by the `req_cfp_slot( )` method.

The paragraphs above describe the sequence of steps that a mobile host executes to be allocated a CFP slot when they do not already have a slot. If a mobile host does have a CFP slot allocated to it, then the mobile host can simply invoke the TBMAC method

The main difference between the sequence of steps in this allocation and the pre-


```
boolean mgr_notify_cfp_alloc( int slot_ref, int address,
                             double delivery_time )
```

Fig. 4.11: Notify slot manager of the allocation of a slot

```
err monitor_cfp_alloc( int slot_ref )
```

Fig. 4.12: Monitor the allocation of a slot

vious one is that the mobile host where the request originates already has a CFP slot. Again, the TBMAC protocol assumes that the slot manager making the invocation has considered the admission control and quality of service implications of making such a request. Therefore, no subsequent intervention is possible on the part of any other slot manager.

4.3.4 Deallocating a slot

The sequence of steps executed to deallocate a CFP slot, illustrated in figure 4.15, is similar to the steps used when requesting a CFP using a previously allocated slot. In both cases, a higher layer, typically a slot manager, initiates the execution. To deallocate a CFP slot, the higher layer calls the TBMAC method

The `slot_ref` parameter to this method is a reference to an allocated CFP slot, previously passed to the caller of the method when the slot was originally allocated.

If the `slot_ref` parameter does not reference an allocated CFP slot or the mobile host on which the method invocation occurs does not have any CFP slots allocated, then the TBMAC protocol ignores the invocation of this method and returns an error. Otherwise, the TBMAC protocol inserts an item into the atomic multicast hash table corresponding to the deallocation request and then piggybacks this information on each of its transmissions.


```
err alloc_cfp_slot( int * slot_ref, int req_type, int address
                  int addition_info )
```

Fig. 4.13: Allocate a slot (assuming caller already has a slot allocated)

```
err dealloc_cfp_slot( int slot_ref )
```

Fig. 4.14: Deallocate a slot (assuming caller already has a slot allocated)

When the TBMAC protocol on a mobile host receives a deallocation request message, it inserts a corresponding item into its local hash table. Note that when the TBMAC protocol receives such a deallocation message, it does not inform a higher layer of the deallocation occurring. This approach of not informing a higher layer is similar to the approach taken when a request to allocate a CFP slot is received in a CFP slot (described in section 4.3.3). Again, the TBMAC protocol assumes the slot manager where the deallocation request originated has taken into account any quality of service considerations before beginning the deallocation.

When the delivery time of the deallocation request arrives, the TBMAC protocol executes a similar set of actions as when the delivery time of an allocation request arrives. However, the TBMAC protocol calls the following method of the slot manager after it updates the allocation of CFP slots:

4.3.5 Allocating/Deallocating an inter-cell slot

The allocation and deallocation of an inter-cell slot uses a similar approach to the allocation and deallocation of a CFP slot. The main difference being that agreement is being sought across two adjoining cells instead of within one cell.

To request the allocation of an inter-cell slot between two adjoining cells, a slot manager invokes the previously described method `alloc_cfp_slot( )`. The parameters to

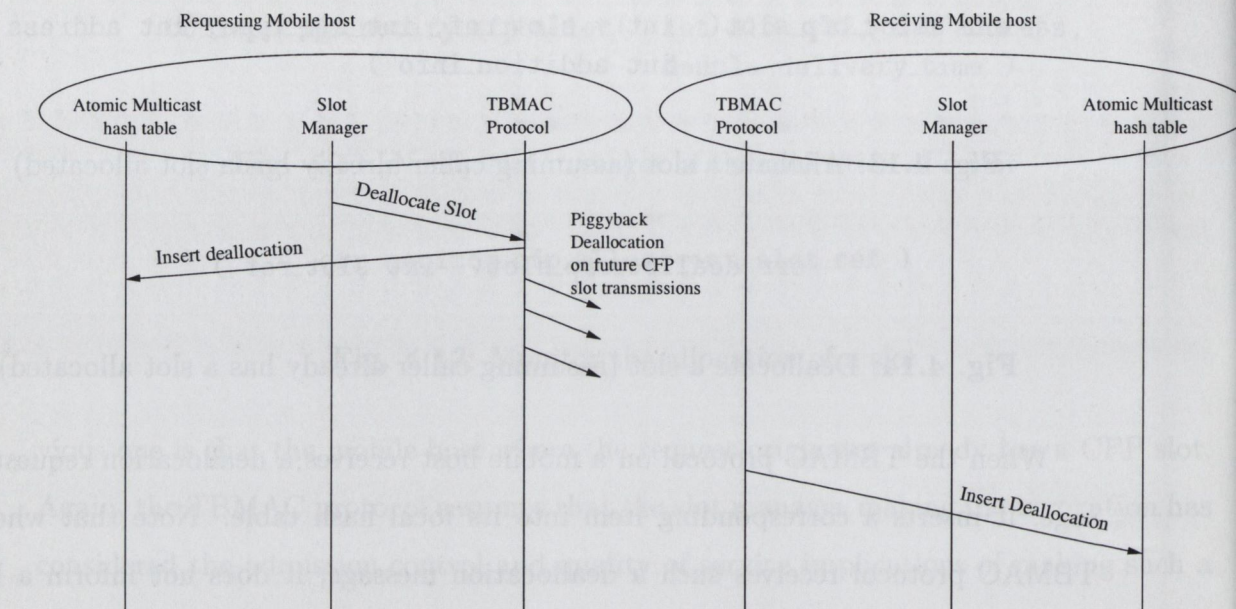


Fig. 4.15: Requesting the deallocation of a CFP slot

```

boolean mgr_notify_cfp_dealloc( int slot_ref, int address,
                                double delivery_time )
  
```

Fig. 4.16: Notify slot manager of the deallocation of a slot

this method are changed to reflect that an inter-cell slot is being requested.

Firstly, the req-type parameter is changed from a CFP slot request to one of three possible inter-cell requests depending on the type of inter-cell communication requested. The three possibilities for this inter-cell communication are communication from the current cell to the neighbouring cell, from the neighbouring cell to the current cell or for communication in both directions. The additional_info parameter is set to the number of the neighbouring cell with which communication is requested. The remaining parameters to the alloc_cfp_slot() method are left unchanged.

To request the deallocation of an inter-cell slot, the dealloc_cfp_slot() is invoked

by a higher layer above TBMAC.

On receiving a request to allocate or deallocate an inter-cell slot, the TBMAC protocol inserts a corresponding item into the hash table of the atomic multicast protocol and then piggybacks the information related to the inter-cell slot request on each of its CFP slot transmissions. If the receiving mobile host has any inter-cell slots for communication with the neighbouring cell with which inter-cell communication is requested, then the receiving mobile host includes this inter-cell request in each of these inter-cell slots.

The delivery delay of the request is increased to reflect the fact that agreement is now being sought across two adjoining cells. Within a single cell, the delivery delay is two TBMAC rounds. This delivery delay is increased to three TBMAC rounds for inter-cell allocation and deallocation requests. Once the delivery time of either of these requests arrive, the corresponding notification method, `mgr_notify_cfp_alloc()` or `mgr_notify_cfp_dealloc()`, is called to inform the slot manager of the allocation or deallocation of the inter-cell slot.

4.3.6 Atomic agreement

In a cell, the TBMAC protocol uses an atomic multicast protocol to reach agreement on the management of slots. To ensure correct functioning of the atomic multicast protocol, the mobile hosts are required to agree on parameters to the atomic multicast protocol. The main parameter to the atomic multicast protocol is the delay to delivery, Δ , which is the duration of time that a message is delayed before delivering the message.

In the TBMAC protocol, there are two distinct values of the Δ parameter. The first value corresponds to when the atomic multicast protocol is used for the allocation and deallocation of CFP slots. The second value is used to allocate and deallocate inter-cell slots. Thus, the second value is typically larger in size than the first due to the longer time needed to reach agreement between cells than within a single cell.

To modify and retrieve the two possible values for the Δ parameter, the following two methods are used. The type parameter to each of these methods specifies whether the intra-cell or inter-cell Δ is being modified or retrieved.

```
err set_delta( int type, double delivery_delay )
double get_delta( int type )
```

Fig. 4.17: Initialise/Retrieve the delay to delivery of the TBMAC protocol

Layers above TBMAC (provided a CFP slot is already allocated) can also use the atomic multicast protocol to provide total ordering on messages that they wish to transmit within a cell. A higher layer initiates an atomic multicast protocol using the following method:

```
err atomic_broadcast( char * buf, int buf_size )
```

The first parameter, `buf`, is the data to be sent using the atomic multicast protocol and the second parameter is simply the size of this data. This data is stored by the TBMAC protocol in its local hash table and then copied in to each CFP slot transmission until the delivery time of the data arrives. By copying the data into each CFP slot, the maximum size of the data cannot be larger than the payload of a CFP slot. The delivery delay, Δ , for this message corresponds to the delivery delay for the allocation/deallocation of a CFP slot.

When the delivery time of the atomic broadcast arrives, the TBMAC protocol notifies the higher layer by calling the following method:

4.3.7 Centralised vs Distributed slot management

The operation of the TBMAC protocol described above is designed to facilitate either centralised or distributed management of CFP slots.


```
err mgr_notify_atomic_delivery( int buf, int buf_size )
```

Fig. 4.18: Notify slot manager of delivery of atomic broadcast

In centralised slot management, one mobile host is responsible for the allocation and deallocation of CFP slots (note that this excludes the deallocation of failed CFP slots as discussed in section 3.3.3). A simple centralised slot management approach would be to order, based on host address, the hosts that have a CFP allocated and to then assign the slot management task to the host with the smallest (or largest) address. This assignment could then be reviewed by each mobile host at the end of each CFP.

The simplicity of this approach is a result of the mobile hosts not needing to elect the central slot manager. The decision as to which mobile host should be the centralised slot manager is done deterministically and the procedure is agreed before the mobile hosts start. A more elaborate election of a slot manager is also possible but would require the mobile hosts with CFP slots to exchange a sequence of messages. Some example possible leadership election algorithms that could be used include [72], [73], [14], [28] and [34]. Once a leader is elected, the remaining mobile hosts would then monitor the status of this leader.

In the centralised slot manager approach, the mobile host assigned the slot manager role would be the only mobile host to respond positively to the `mgr_notify_cfp_request()` invocation from the TBMAC protocol (see section 4.3.3) while all other mobile hosts respond negatively to the invocation. The disadvantage of the centralised slot management approach is that the centralised slot manager must correctly receive the request for a slot from a mobile host which then results in the atomic multicast protocol being executed.

The simplest form of distributed management of slots is when there is no slot management layer present above the TBMAC protocol present. In this instance, the


```
err send( int slot_ref, Packet * pkt )
```

Fig. 4.19: Queue a packet for transmission by the TBMAC protocol

TBMAC protocol simply proceeds with any allocation or deallocation of CFP slots. Requests for a CFP slot will fail if all the slots in the static CFP are already allocated.

A variant of this simple approach is a slot manager that simply allows all requests for CFP slots to proceed. Currently, this is the only slot manager implemented in the NS2 simulator.

A more complicated form of distributed slot management would require the mobile hosts to agree on whether or not to proceed with each allocation or deallocation of a CFP slot. For example, when a slot manager running on any mobile host is notified, via `mgr_notify_cfp_request()` of a request for a CFP slot, the slot manager would respond negatively to the invoking TBMAC layer. The slot manager could then reach a distributed agreement with the other slot managers in the cell on whether or not to proceed with the request, possibly, using `atomic_broadcast()`.

4.4 Communication

Once a mobile host has a cfp slot allocated to it, the mobile host can then transmit messages in its slot every CFP period using the following method:

This method queues the packet, passed as a parameter, for transmission in the CFP slot referenced by the `slot_ref` parameter.

There are three options for the reception of messages. TBMAC can simply pass the received packet to the higher layer for processing, pass the received packet directly to the slot manager or do both (passing a copy of the packet to the slot manager). Messages can either be queued by the TBMAC layer, or a higher layer can register a call back to be notified when a message arrives.


```
void recv( Packet * pkt )
```

Fig. 4.20: Pass packet to higher layer in protocol stack

The first option uses the following NS2 related method to pass a packet to the next layer in the protocol stack.

The TBMAC protocol requires the higher layer slot manager to enable notifications of the reception of packets using the `enable_pkt_notify( )` method below.

```
err enable_pkt_notify(  boolean enable,  
                       boolean make_copy )
```

Fig. 4.21: Enable copy of packet to be passed to slot manager

The second parameter to this method specifies whether the packet should be copied before being passed to both the next layer in the protocol stack and the slot manager.

Once packet reception notifications are enabled, the TBMAC protocol calls the following method of the previously registered instance of the SlotMgt class

```
boolean mgr_notify_pkt(  int slot_ref,  
                        Packet * pkt )
```

Fig. 4.22: Pass received packet to slot manager

Allowing each received packet to be copied and passed to both the next layer up the protocol stack and also to a slot manager allows existing protocols (at the routing and higher layers) to operate without any specific knowledge of the allocation of TBMAC slots.

4.5 Summary

In this chapter, an overview of the implementation of the TBMAC protocol in the NS2 network simulator was given. The beginning of the chapter covered how the TBMAC protocol is initialised and presented the main data structures used in the implementation. The chapter then described the interaction between the TBMAC protocol and a higher slot management layer, including the allocation and deallocation of CFP slots and the transmission of packets in allocated slots.

Chapter 5

Evaluation

Chapter 1 identified the contribution of this thesis as the design of a new MAC protocol providing time-bounded medium access control in multi-hop wireless ad hoc networks. This protocol is called TBMAC. TBMAC provides, with known probabilities, a set of time bounds for access to the wireless medium that depend on the current mode of the protocol.

In the next section, an overview of the general approach used in this evaluation is given in section 5.1. A description of the environment in which the experiments were carried out is then given in section 5.2. Following this description, the set of experiments executed is presented in section 5.3. Both the expected and actual results of these experiments are covered in section 5.4. This section also discusses other relevant results of these experiments related to the TBMAC protocol such as the detection of failed slots and the failure assumptions of the atomic multicast protocol. Finally, the chapter concludes with a summary.

5.1 Approach

Chapter 3 described the design of the TBMAC protocol and highlighted the time bounds provided by TBMAC. In total, chapter 3 covered five different time bounds of the TBMAC protocol. However, three of these five time bounds can be considered as depending on whether two main time bounds are achieved or not.

The two main time bounds were, firstly, the time bound for the allocation of a CFP slot in a non-empty cell to a mobile host that does not already have a CFP slot allocated and, secondly, the time bound for the allocation of at least one CFP slot at the start of the bootstrapping protocol. For the remainder of this chapter, this first time bound will be called the non-empty cell time bound while the second time bound will be called the empty cell time bound. The first time bound corresponds to a mobile host choosing a slot in the CP while the second time bound corresponds to a mobile host choosing at least one distinct CFP slot in a randomly generated list of slots at the start of the bootstrapping protocol. The probabilities for these two time bounds were derived mathematically in section 3.4 at the end of Chapter 3.

The primary goal of this evaluation is to experimentally validate these derived probabilities by obtaining a set of corresponding statistics from the set of experiments executed. The obtained statistics are then compared to their expected probabilities.

In addition, the effect of factors such as the number of inconsistently allocated CFP slots, the number of mobile hosts that fail to transmit in or use an allocated CFP slot and the detection rate of unused CFP slots by the TBMAC protocol is evaluated.

The evaluation of the two time bounds and additional factors, described above, was carried out using the implementation of the TBMAC protocol described in chapter 4. This implementation of the TBMAC protocol used the NS2 network simulator. It is important to understand how the NS2 simulator operates, this will in turn allow a better understanding of how the evaluation was carried out. Firstly, the NS2 simulator requires that the maximum number of mobile hosts in an experiment is specified before

the experiment begins. NS2 also requires the duration of the experiment and the movement pattern that the set of mobile hosts in the experiment will follow to be specified before the experiment begins.

When an experiment is executed using the simulator, a trace file is created that contains results corresponding to that experiment. This trace file must then be processed after the experiment has finished executing in the simulator to obtain the relevant statistics for that experiment.

5.2 Environment

The evaluation environment consisted of the implementation of the TBMAC protocol carried out with the NS2 simulator. The version of the NS2 simulator used (before the TBMAC implementation was carried out) was ns-allinone-2.1b9a [2].

The code of the NS2 simulator was compiled on an Intel Pentium III processor with a clock speed of 1GHz and 256MB of memory, running the GNU/Linux distribution Red Hat 7.3 (linux kernel version 2.4.18-3 patched by Red Hat). The compiler used was the default g++ compiler on this operating system (version 2.96.11 patched by Red Hat).

A single sequential run of the set of experiments described in section 5.3 on the above machine lasts between one and two days. To speed up the duration of an experimental run the machine on which the NS2 simulator was compiled was assigned as a central repository for the set of experiments to be executed. The set of experiments were then statically divided and executed by 8 workstations.

The 8 workstations each had similar hardware layout to the machine on which the NS2 simulator was compiled. In contrast to that machine, the operating system installed on each of these workstations was Red Hat version 9, a distribution of the GNU/Linux operating system, running the Red Hat linux kernel 2.4.20-8. Each work-

station locally mounted the directory containing the compiled NS2 simulator binary from the central repository using NFS. Each individual workstation then executed a statically defined sub-set of the experiments described in the next section using the NS2 simulator.

5.3 Experimental Configuration

As outlined in section 5.1, the primary goal of the evaluation was to compare obtained statistics for attaining the two main time bounds of the TBMAC protocol and compare these statistics against their corresponding expected probabilities.

Instead of running a distinct set of experiments for each of the two time bounds, statistics for each of the two time bounds were obtained from running one default set of experiments. Using a default set of experiments reduced the number of experiments that were required to be executed and was an optimisation in terms of the of the duration of time needed to execute these experiments.

The trace files generated from executing this default set of experiments contained the results for each individual statistic. These trace files also contained results for the additional factors (highlighted in section 5.1) also under investigation. Thus the experiments were executed and the trace files were then parsed for the particular statistic of interest.

The configuration of the set of experiments carried out can be described under two distinct headings:

1. Movement pattern of mobile hosts
2. TBMAC protocol parameters

Each of these areas will now be discussed, followed by a summary of the experimental parameters.

5.3.1 Movement pattern of mobile hosts

For all the experiments mobile hosts moved within a single hexagonal cell. In different experiments, the length of one side of this hexagonal cell varied from 125m through 150m to 175m. Thus, the maximum possible distance between any two mobile hosts in the cell varied from 250m to 350m (or the diameters of the three hexagonal cells considered were 250m, 300m and 350m respectively).

The propagation model used in all these experiments was the Two-Ray ground model that comes as a standard component in the NS2 simulator. With this propagation model in the simulator, two mobile hosts within 250m of each other are able to correctly receive each other's transmissions. Thus, the underlying reason for choosing the above cell diameters was to investigate the performance of TBMAC when the assumption of a single hop network within a cell is both true and false.

The number of mobile hosts participating in each experiment was also varied from 25 mobile hosts to 100 mobile hosts in increments of 25. This was to investigate whether or not the number of mobile hosts had an effect on obtaining the time bounds of the TBMAC protocol.

For each experimental run, the full set of mobile hosts to be simulated in the particular experiment were randomly placed in an inactive state on the boundary of the hexagonal cell as illustrated in figure 5.1. This random placement was carried out by first randomly choosing one of the sides of the hexagon cell and then randomly placing the mobile host on this chosen side. A linear congruential pseudo-random number generator [44] was used to implement this random placement.

The activation of an inactive mobile host corresponds to a mobile host arriving into the cell before moving across it as illustrated in figure 5.2. The arrival rate of mobile hosts (or more precisely the activation rate) corresponded to a Poisson process. The parameter to this Poisson process, λ , was varied between 1 and 2 causing on average 1 or 2 mobile hosts to arrive into the cell during each second of the particular experiment.

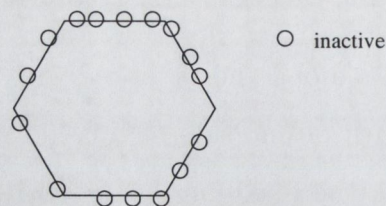


Fig. 5.1: Sample placement of mobile hosts at the start of an experiment

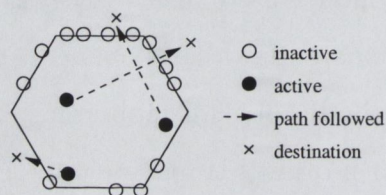


Fig. 5.2: 3 activated mobile hosts in an experiment

Once activated, each mobile host chooses a destination point just outside the cell boundary to which to move and then moving to this destination point at 15m/s. As mobile hosts were randomly placed on the cell boundary and the destination of each host in the experiment is also randomly chosen, the duration of time that a mobile host spends in the cell should be random and therefore it was deemed unnecessary to vary the speed of a mobile host.

On being activated, a mobile host attempts to allocate itself a CFP slot (either by making a Slot Allocation request in the CP or bootstrapping the cell). On obtaining a CFP slot, a mobile host transmits Null messages in this slot until it leaves the cell.

Note that it is possible for a mobile host to request a CFP slot and to then leave the cell before this request is delivered. In this case, the mobile host is regarded as having failed. Otherwise, any mobile host that has a CFP slot allocated always deallocates this CFP slot before leaving the cell.

As highlighted in section 5.1, the duration of each experiment had to be defined be-

fore the experiment was executed. Thus the duration of each experiment corresponded to the amount of time, on average, needed to activate the specified number of mobile hosts given the particular arrival rate. Thus with an arrival rate of 1 mobile host per second and 25 mobile hosts participating in the experiment, an experiment would last 25s whereas an arrival rate of 2 mobile hosts per second and 100 participating mobile hosts results in an experiment lasting 50s.

5.3.2 TBMAC parameters

The sizes of the CFP and CP were both fixed for all experimental runs with the CFP having 100 slots, each of duration 0.005s, and the CP having 40 slots, each of duration 0.0025s. This resulted in a TBMAC round lasting 0.6s. The data rate of the physical layer in the NS2 simulator underlying the TBMAC protocol was 1Mb/s.

As the size of the generated list of the bootstrapping protocol impacts on the probability of the empty cell time bound being obtained. This list size was varied between 2, 3 and 4 to check whether there was a corresponding change in the results for the empty cell time bound.

In both operational and bootstrapping modes, the TBMAC protocol must randomly choose slots in either the CP or the CFP. In the implementation of TBMAC, the selection of these slots was implemented using the same random number generator used to generate the placement of mobile hosts on the cell boundary (see section 5.3.1). Each mobile host running the TBMAC protocol had such a random number generator and it was seeded with the time that this mobile host becomes active.

As described in chapter 3, the Additional Info field of a TBMAC packet header is bounded in size. For the set of experiments, the Additional Info field was fixed in size to contain a maximum of 10 messages of the atomic multicast protocol. Recall from chapter 3 that the Additional Info field is also used to transmit failure indications when a CFP slot is not being used. For the set of experiments executed, an update

requesting the allocation or deallocation of a slot was inserted into the Additional Info field before any message indicating the failure of a slot.

Finally, the TBMAC protocol running on each mobile host did not make any admission control decisions on whether or not to proceed with a request to allocate a CFP slot. When a mobile host receives a request for a CFP slot, the mobile host proceeds to execute the steps required to allocate the requesting mobile host a slot in the CFP.

5.3.3 Summary of parameters

Table 5.1 summarises the variable parameters to the experiments. λ is the arrival rate of mobile hosts into the cell, N is the number of mobile hosts participating in a particular experiment and j is the size of the list generated in the bootstrapping protocol. As noted before, the diameter of the hexagonal cell in the experiments varied in size between 250m, 300m and 350m. The first three columns of table 5.1 can be considered movement related parameters while the last column is related to the TBMAC protocol.

One hundred experiments were generated for each combination of the experimental parameters resulting in 7,200 different experiments.

λ	N	Cell Diameter	j (list size)
1	25	250	2
2	50	300	3
	75	350	4
	100		

Table 5.1: Summary of values of experimental parameters investigated

5.4 Experimental Results

This section presents the results of the experiments carried out. In section 5.4.1, the expected probabilities for the time bounds of the TBMAC protocol are reviewed. This is followed in section 5.4.2 by a comparison with the values obtained from the set of experiments described in 5.3. The results of the experiments for the secondary factors, outlined in section 5.1 follows in section 5.4.3.

5.4.1 Expected Time Bound Probabilities

The probabilities associated with the two time bounds corresponding to the cases of non-empty and empty cells will now be covered.

Non-empty cell

For the experimental parameters described in section 5.3, a graph of the expected probability of hosts choosing a distinct slot in the CP is given in figure 5.3 with the x axis denoting the number of hosts (k). For each value of k , the y axis gives the expected probability of k hosts choosing a distinct CP slot.

Combining the probability values in figure 5.3 with the probability of a host arriving in the cell according to a Poisson process, modelling arrival rates, gives figure 5.4. The two graphs in figure 5.4 correspond to λ varying between 1 and 2. The time parameter, t , to the Poisson process corresponds to the duration of a TBMAC round (in these experiments t equals 0.6s). The peak in figure 5.4 for the λ equal to 2 graph at k equal to 1 is due to the long run average of the Poisson process being centered around 2.

Summing each of the probabilities in figure 5.4 for a particular arrival rate (graph) gives the probability of a host choosing a distinct slot in the CP for that arrival rate. The probability of a mobile host choosing a distinct slot in the CP corresponds to the probability of the time bound being obtained in the non-empty cell. Each of these

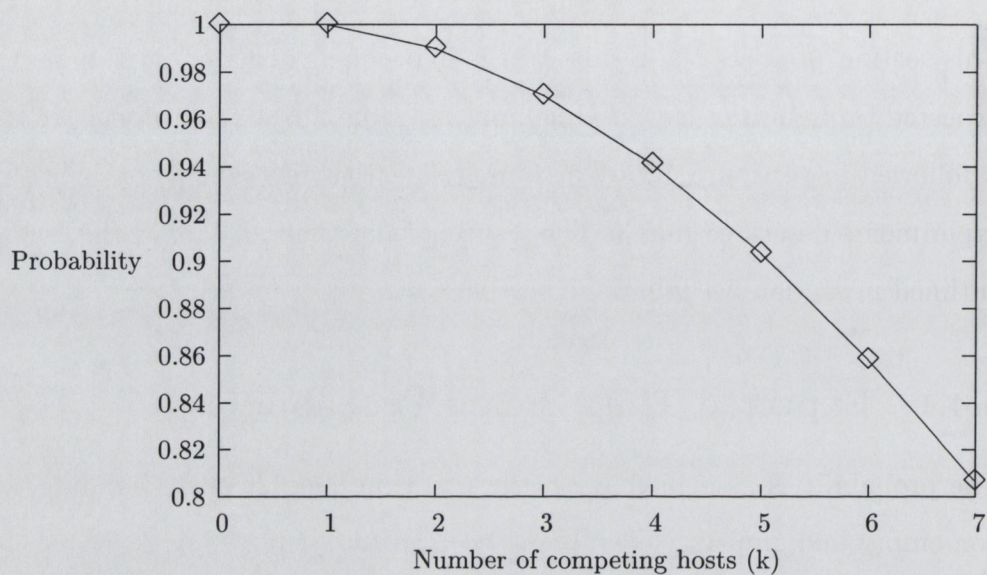


Fig. 5.3: Expected probabilities of a host choosing a distinct CP slot

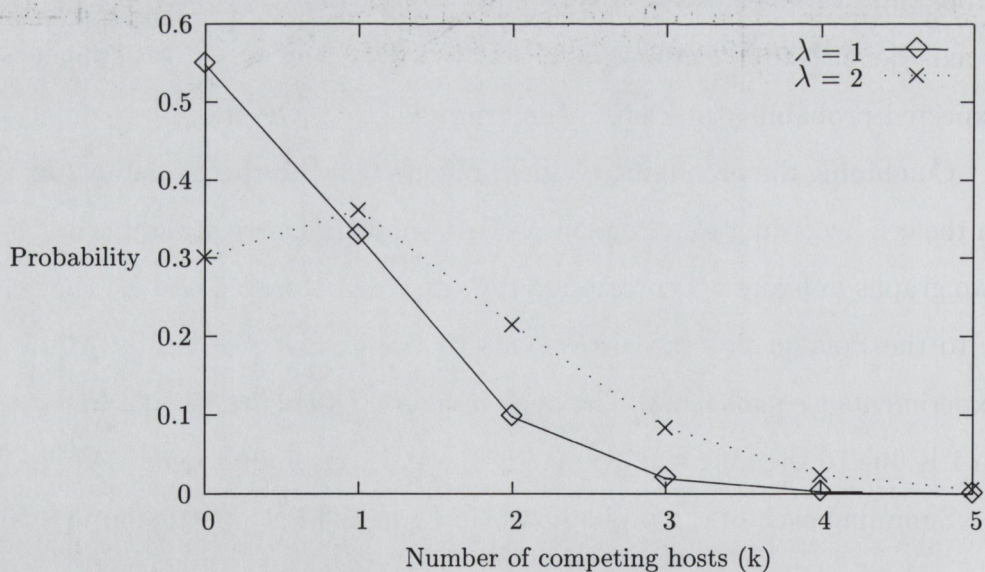


Fig. 5.4: Expected probabilities of one or more hosts arriving within t seconds and each of these hosts choosing a distinct CP slot

summed probabilities are shown in table 5.2. As would be expected, the probability of choosing a distinct slot is higher with a smaller arrival rate as there is, on average, less mobile hosts competing with each other.

λ	P(host choosing a distinct CP slot)
1	0.99817561
2	0.991610278

Table 5.2: Expected probability of a host choosing a distinct slot in the CP

Empty cell

Similarly, it is possible to calculate the expected probability of the empty cell time bound being obtained.

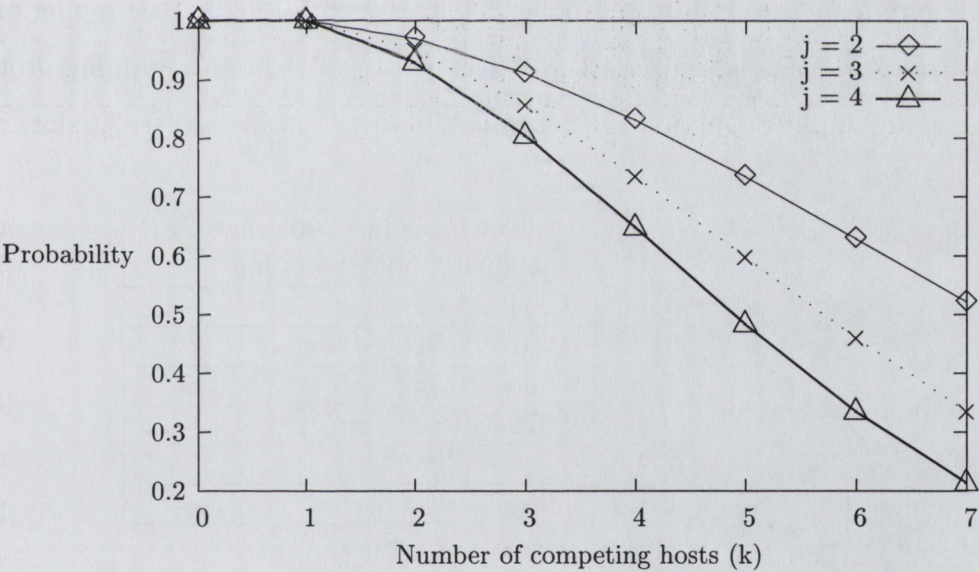


Fig. 5.5: Expected probabilities of a host choosing at least one distinct CFP slot

A sample of the expected probabilities of a single host choosing at least 1 distinct slot at the start of the bootstrapping protocol is given in Figure 5.5. The set of graphs

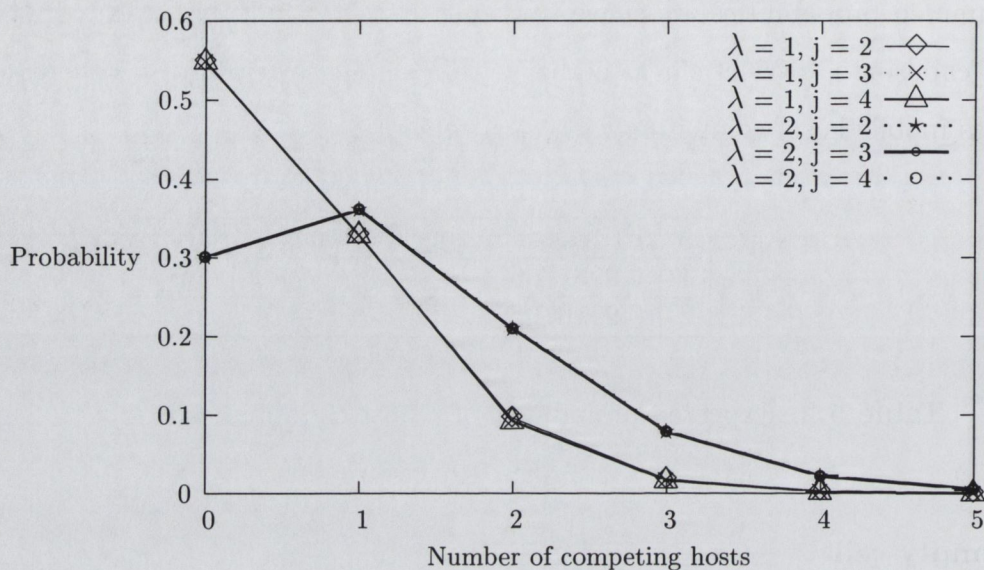


Fig. 5.6: Expected probabilities of one or more hosts arriving within t seconds and each of these hosts choosing at least one distinct CFP slot

in figure 5.5 correspond to different list sizes used in the bootstrapping protocol. Note that by increasing the size of the generated list of the bootstrapping protocol, there is a corresponding reduction in the probability of choosing a distinct slot.

λ	j	P(host choosing a distinct CFP slot during bootstrapping)
1	2	0.994712632
1	3	0.991341151
1	4	0.988086133
2	2	0.979321079
2	3	0.966647006
2	4	0.954759287

Table 5.3: Expected probability of a host choosing a distinct slot in the CFP

Similarly to the non-empty cell time bound, it is possible to combine the probability values in figure 5.5 with the Poisson process, modelling arrival rates, giving the 6 graphs in figure 5.6. As can be seen from figure 5.6, the arrival rate dominates these probability

values with the three graphs for $\lambda = 1$ (where $j = 2, 3, 4$) coinciding. Similarly, for the three graphs when $\lambda = 2$.

Summing each of these combined probabilities gives the probability of a host obtaining the empty cell time bound and is shown in table 5.3. As in table 5.2, the trend of the probabilities in the table is to decrease as the arrival rate increases. The probabilities also decrease as the size of the list of generated slots in the bootstrapping protocol increases.

5.4.2 Obtained Time Bound Statistics

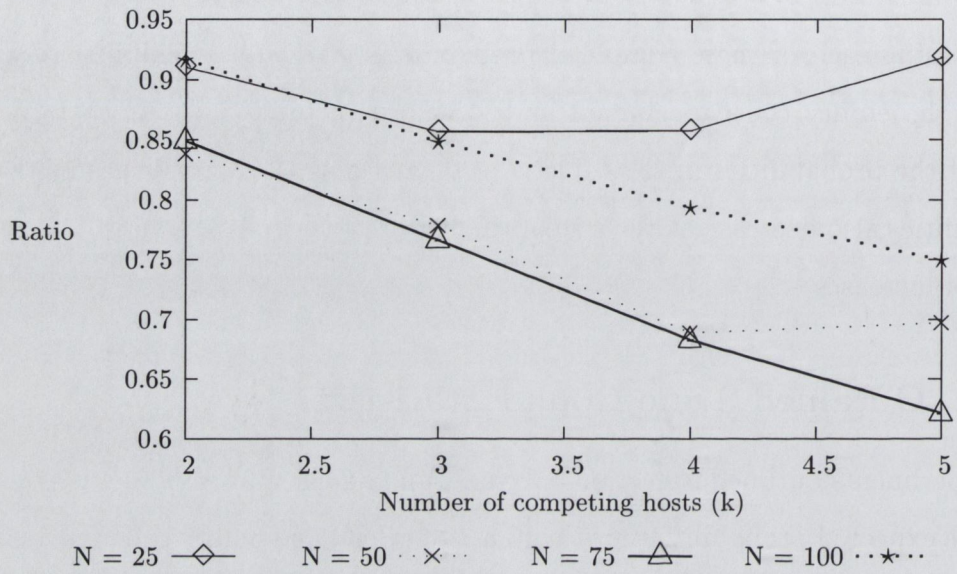
The experiments outlined in section 5.3 were run and the statistics required to compare with the expected probabilities for the non-empty cell and empty cell time bounds were collected.

Non-empty cell

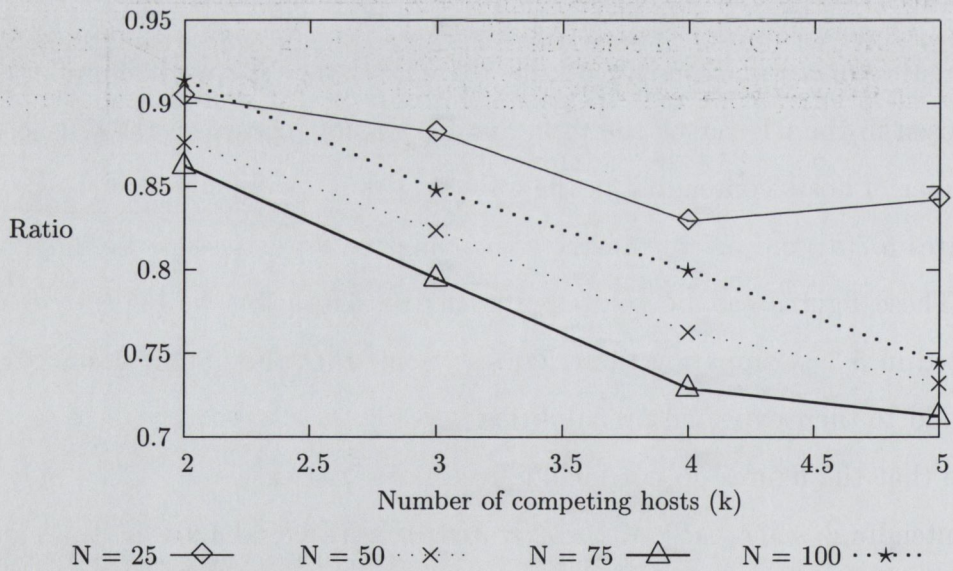
The statistics for the non-empty cell time bound was the ratio of the number of distinct slots chosen in the CP versus the total number of slots chosen in the CP according to the number of hosts contending in the CP.

Figures 5.7(a) and 5.7(b) illustrate the collected statistics for the different arrival rates. These figures can be compared with the figure for the expected probability illustrated in 5.3. Comparing these figures we see that the obtained statistics do not correspond to their expected probabilities.

Note that the figures do not include results for the case $k = 1$ (i.e., only 1 mobile host contending) since each of these statistics were equal to 1 in the experiments corresponding to the expected probability. The results for values of k greater than 5 are also not included in these figures due to the relatively small number of total occurrences of 5 or more hosts in the cell contending in the CP. Note also that the results in these figures consist of the averages of the experiments where the list size of



(a) $\lambda = 1$



(b) $\lambda = 2$

Fig. 5.7: Obtained statistics for a host choosing a distinct CP slot

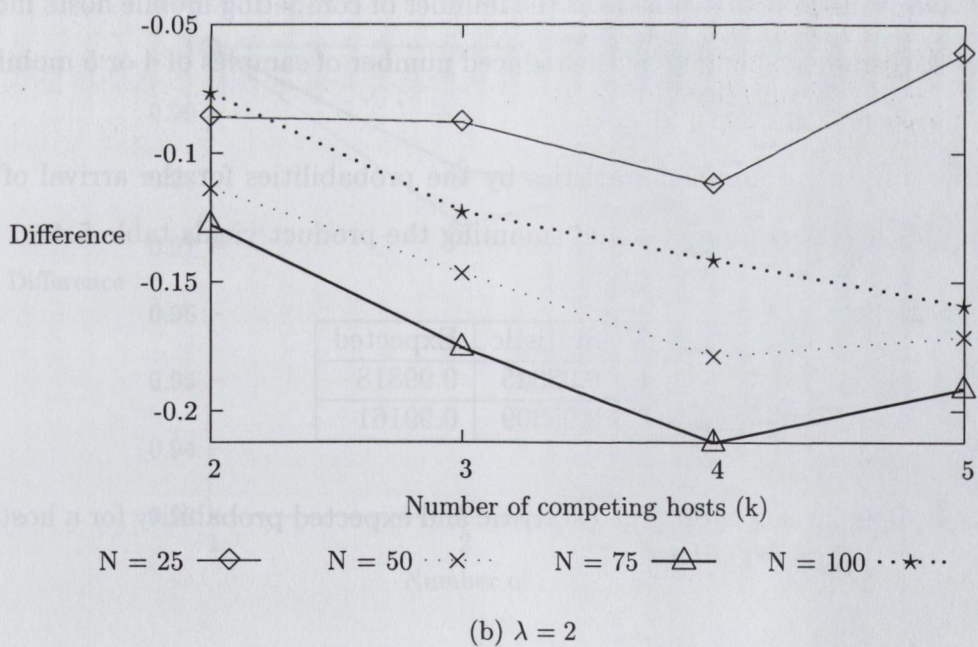
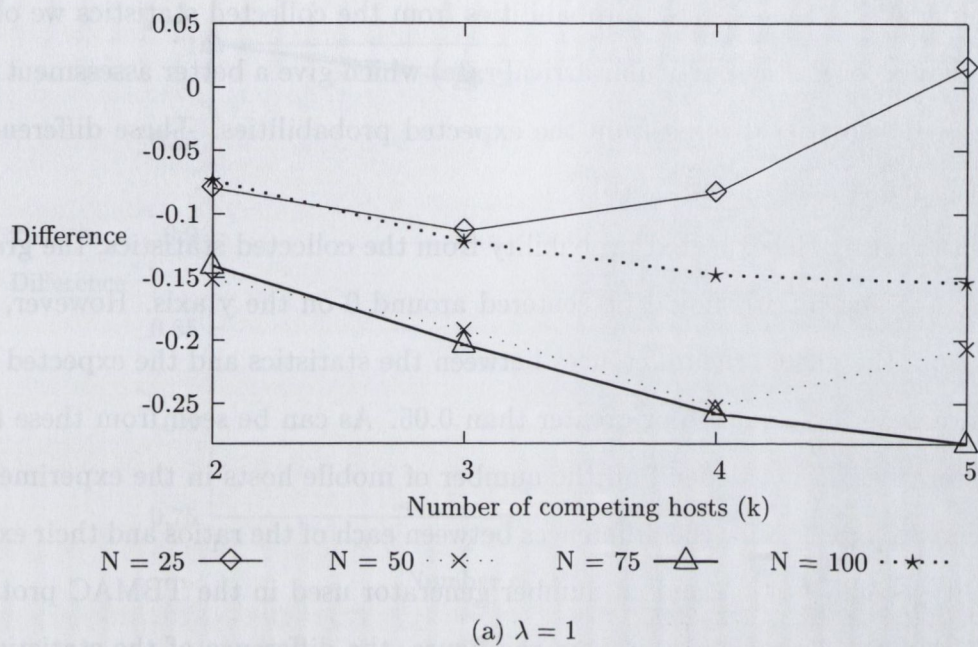


Fig. 5.8: Difference between obtained statistics and expected probabilities for a host choosing a distinct CP slot

the bootstrapping protocol is 2, 3 and 4.

By subtracting the expected probabilities from the collected statistics we obtain a set of differences (for the particular arrival rate) which give a better assessment of how the collected statistics diverge from the expected probabilities. These differences are graphed in figures 5.8(a) and 5.8(b).

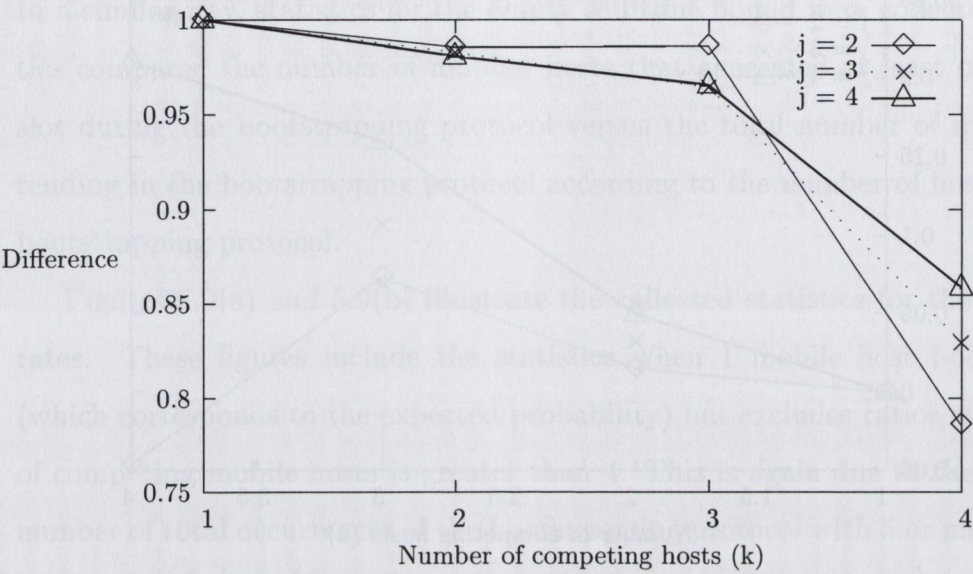
By subtracting the expected probability from the collected statistics, the graphs in figures 5.8(a) and 5.8(b) should be centered around 0 on the y axis. However, as can be seen from the graph, the differences between the statistics and the expected probabilities are generally equal to or greater than 0.05. As can be seen from these figures, these differences do not depend on the number of mobile hosts in the experiment.

One possible reason for the differences between each of the ratios and their expected probabilities could be the random number generator used in the TBMAC protocol to choose CP slots. As can be seen from the figure, the difference of the statistics from the expected probabilities increases as the number of competing mobile hosts increases. This increase can be explained by the reduced number of samples of 4 or 5 mobile hosts contending with each other.

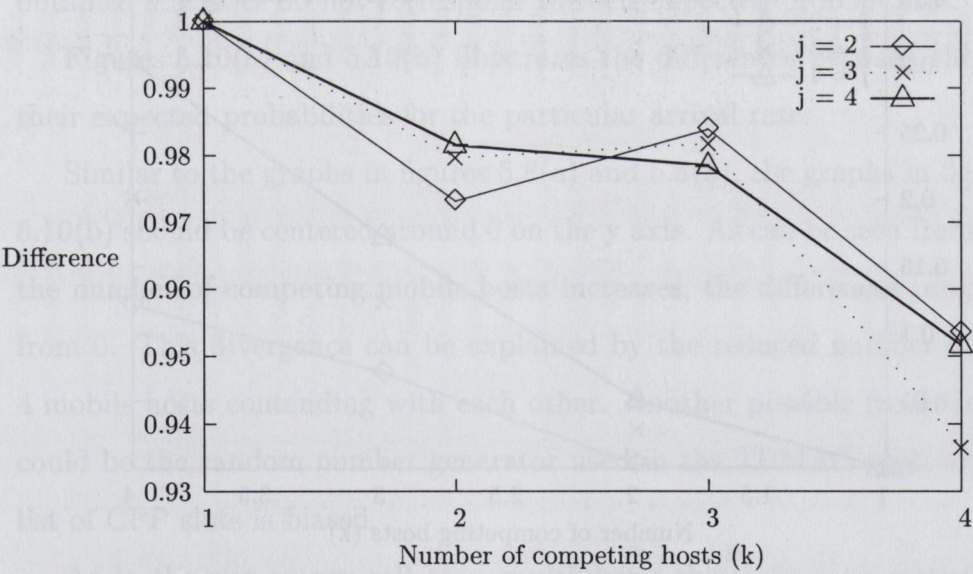
Multiplying these obtained statistics by the probabilities for the arrival of a host according to the Poisson process and summing the product yields table 5.4.

λ	Statistic	Expected
1	0.98345	0.99818
2	0.95309	0.99161

Table 5.4: Comparison of obtained statistic and expected probability for a host choosing a distinct slot in the CP

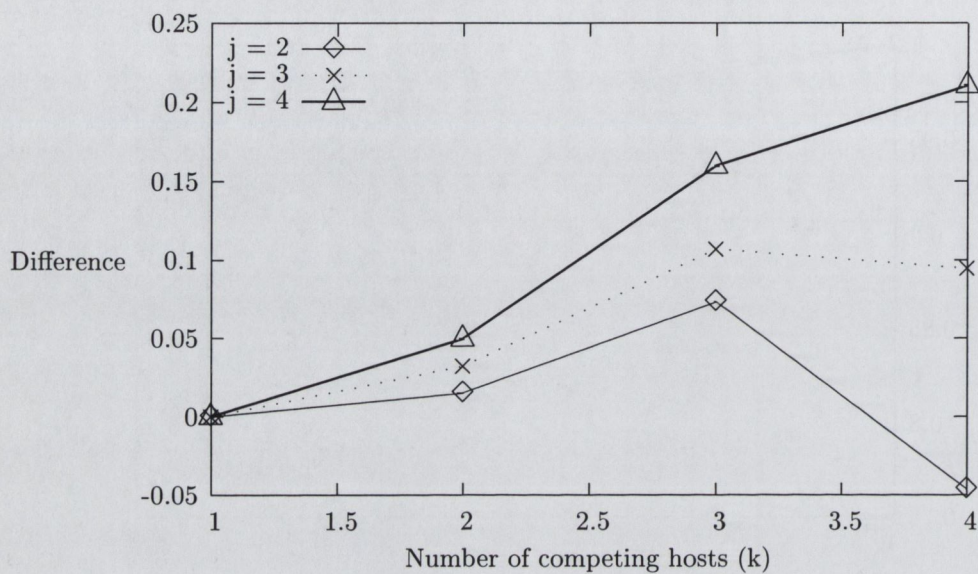


(a) $\lambda = 1$

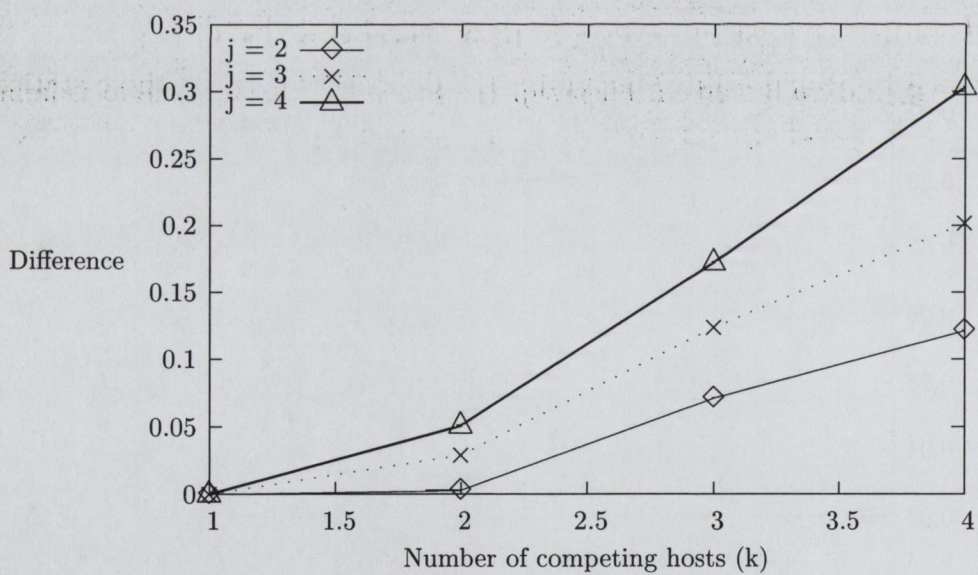


(b) $\lambda = 2$

Fig. 5.9: Obtained statistics for a host choosing at least one distinct CFP slot



(a) $\lambda = 1$



(b) $\lambda = 2$

Fig. 5.10: Difference between obtained statistics and expected probabilities for a host choosing at least one distinct CFP slot

Empty cell

In a similar way, statistics for the empty cell time bound were collected. These statistics compared the number of mobile hosts that generated at least one distinct CFP slot during the bootstrapping protocol versus the total number of mobile hosts contending in the bootstrapping protocol according to the number of hosts executing the bootstrapping protocol.

Figures 5.9(a) and 5.9(b) illustrate the collected statistics for the different arrival rates. These figures include the statistics when 1 mobile host bootstraps the cell (which corresponds to the expected probability) but excludes ratios where the number of competing mobile hosts is greater than 4. This is again due to the relatively small number of total occurrences of the bootstrapping protocol with 5 or more mobile hosts.

The statistics in these figures can also be compared with their corresponding expected probability illustrated in 5.3. Again, comparing these figures we see that the obtained statistics do not correspond to their expected probabilities.

Figures 5.10(a) and 5.10(b) illustrates the differences between these statistics and their expected probabilities for the particular arrival rate.

Similar to the graphs in figures 5.8(a) and 5.8(b), the graphs in figures 5.10(a) and 5.10(b) should be centered around 0 on the y axis. As can be seen from these figures, as the number of competing mobile hosts increases, the differences progressively diverge from 0. This divergence can be explained by the reduced number of samples of 3 or 4 mobile hosts contending with each other. Another possible factor in this divergence could be the random number generator used in the TBMAC protocol to generate the list of CFP slots is biased.

As in the non-empty cell case, multiplying these obtained statistics by the probabilities for the arrival of a host according to the Poisson process and summing the product yields table 5.5.

λ	j	Statistic	Expected
1	2	0.99725	0.99471
1	3	0.99669	0.99134
1	4	0.99661	0.98809
2	2	0.98381	0.97932
2	3	0.98459	0.96665
2	4	0.98510	0.95476

Table 5.5: Comparison of obtained statistic and expected probability of a host choosing a distinct slot in the CFP

5.4.3 Other Results

The effect of factors such as the number of inconsistently allocated CFP slots, the number of mobile hosts that fail to transmit in or use an allocated CFP slot and the detection rate of unused CFP slots by the TBMAC protocol was also evaluated using the set of experiments described in 5.3. The experimental results for each of these factors will now be discussed.

Inconsistent Allocation / Deallocation

An important component of the TBMAC protocol is the atomic multicast protocol used to allocate and deallocate CFP slots. A question related to the atomic multicast protocol is whether the allocation / deallocation of CFP slots is carried out consistently across all the active hosts in the cell or not. This question relates to whether the assumptions of the atomic multicast protocol are valid.

Figures 5.11(a) and 5.11(b) gives the ratio of slot allocations and deallocations that were delivered inconsistently across the mobile hosts in the cell over the total number of slot allocations and deallocations delivered. From figures 5.11(a) and 5.11(b), inconsistency increases as the size of the cell increases. A reason for this rise is that as the size of the cell increases, the assumption that TBMAC makes about the size of the cell

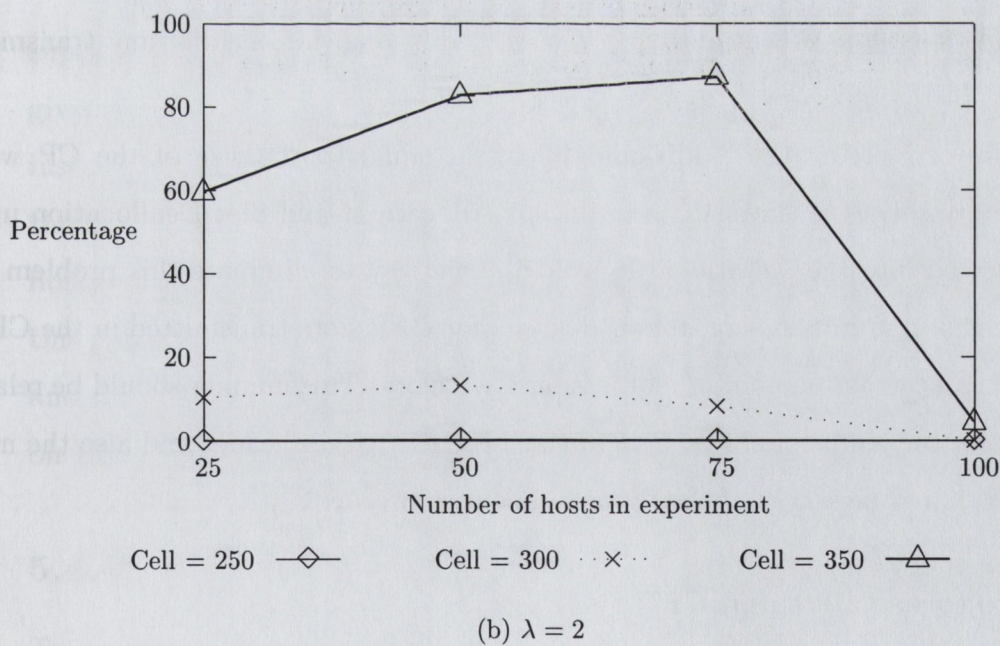
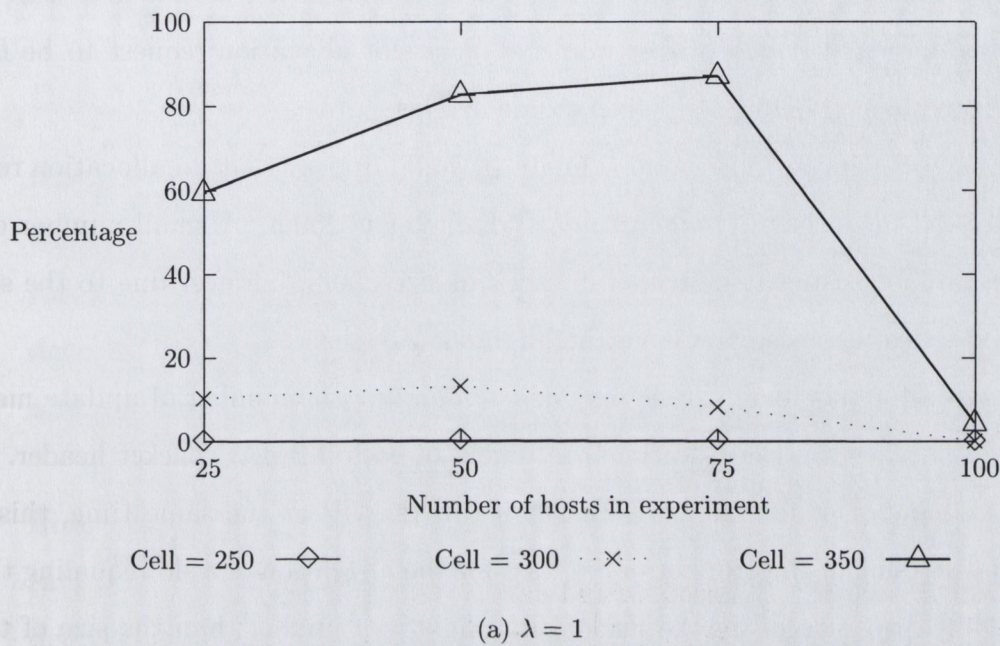


Fig. 5.11: Percentage of inconsistently delivered slot allocation requests

corresponding to a single hop network becomes less and less valid. A solution to this problem would be to increase the delay to delivery, Δ , of the atomic multicast protocol. This increased delivery time would allow a slot allocation request to be flooded throughout the cell before its delivery time arrives.

Looking at figures 5.11(a) and 5.11(b), it would appear that no allocation requests are delivered inconsistently when the cell diameter is 250m. A small number of allocations are inconsistently delivered in a cell of size 250m, however due to the scale of the y axis, this inconsistency is not highlighted.

A reason for this inconsistency is due to bounding the number of update messages that can be placed in the Additional Info field of each TBMAC packet header. As an arbitrary number of mobile hosts can arrive in the CP at the same time, this could potentially result in the number of requests for the allocation of a slot equaling the size of the CP. If the size of the Additional Info field was smaller than the size of the CP then the possibility of inconsistency is increased due to mobile hosts within the cell not including the Slot Allocation request in the Additional Info field of any transmissions in their CFP slots.

However, setting the Additional Info field equal to the size of the CP will not completely solve the problem as both Slot Allocations and Slot Deallocation updates are inserted into the Additional Info field. One way to eliminate this problem would be to perform admission control on requests for CFP slots transmitted in the CP, only accepting a certain number of requests for CFP slots. This number should be related to the size of the Additional Info field of the TBMAC packet header and also the number of outstanding messages of the atomic multicast protocol.

Failure to use allocated CFP slot

Another important statistic from the experiments is the percentage of mobile hosts in the experiments that leave the cell before they are allocated a CFP slot (these were

labeled failed hosts in section 5.3). As can be seen from figures 5.12(a) and 5.12(b), the number of such mobile hosts varies from 1/4 (25.47) of all hosts down to 1/20 (5.09).

Detection of unused CFP slot

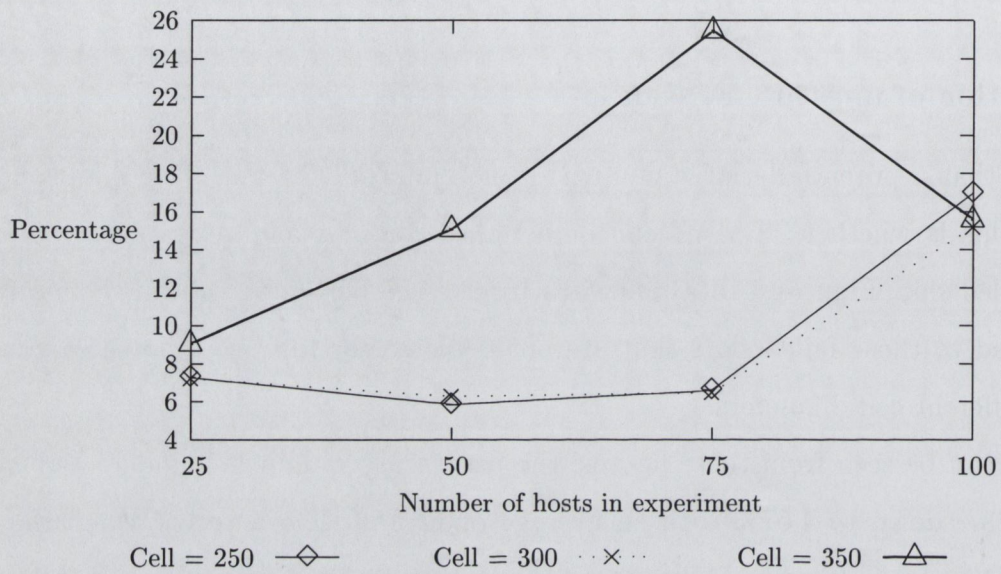
Given that a potentially large number of mobile hosts can fail. One question arising from this is whether TBMAC can detect these failures and subsequently deallocate the failed slot. Figures 5.13(a) and 5.13(b) give the percentage of failed slots that are detected to those failed slots that are not detected for the two different arrival rates and different cell diameters.

As can be seen from these figures, the percentage of failed slots detected can vary from 23% down to 4.67% with an average of 12% of slots detected when the arrival rate is 1 and 6.5% detected when the arrival rate is 2. This gives an average slot failure detection rate of 9.25% over all the experiments.

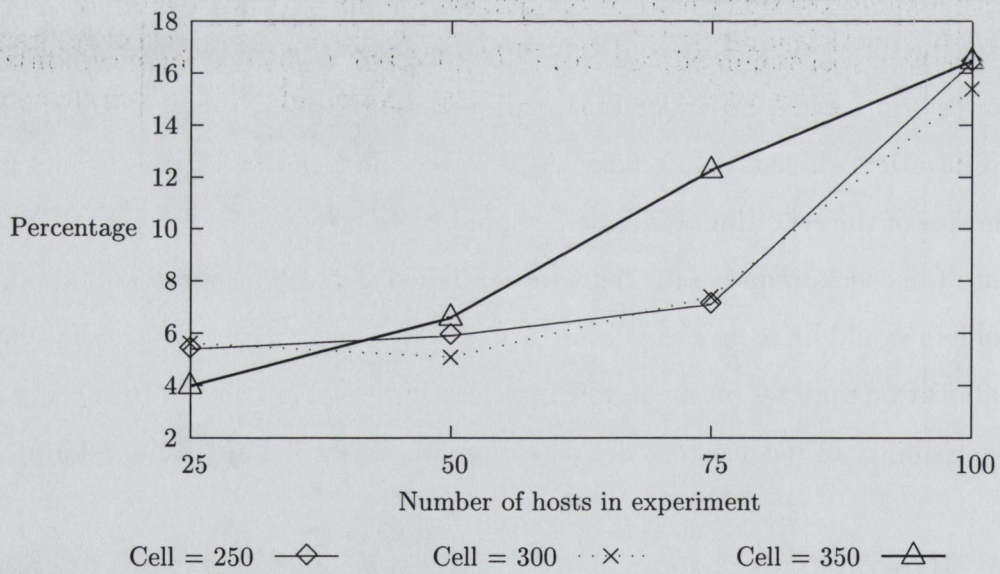
As highlighted in section 5.3, the maximum number of atomic broadcast messages supported by TBMAC was 10 and that updates indicating the failure of the host are given lower priority in comparison to slot allocation or deallocation updates when being inserted into the Additional Info field. Again, the values in these figures would indicate that the size of the Additional Info field should be tailored to the arrival rate of mobile hosts into the cell to improve the detection of failed slots. Another possible solution to this problem would be to give the failure updates higher priority over the slot allocation and deallocation updates of the atomic multicast protocol but this in turn could impact on the possibility of inconsistent delivery of a slot allocation or deallocation request.

5.4.4 Remarks

To conclude this section, the evaluation highlighted a number of artefacts related to the implementation of the TBMAC protocol. Firstly, the rate at which requests for CFP slots can be delivered consistently by the atomic multicast protocol depends on the



(a) $\lambda = 1$



(b) $\lambda = 2$

Fig. 5.12: Percentage of hosts that fail to use their slot

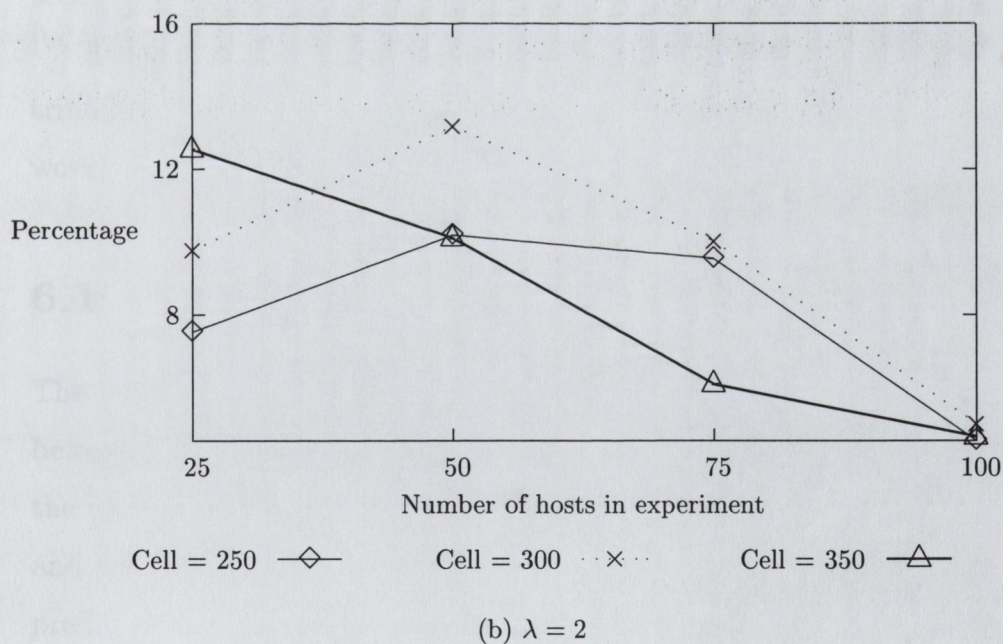
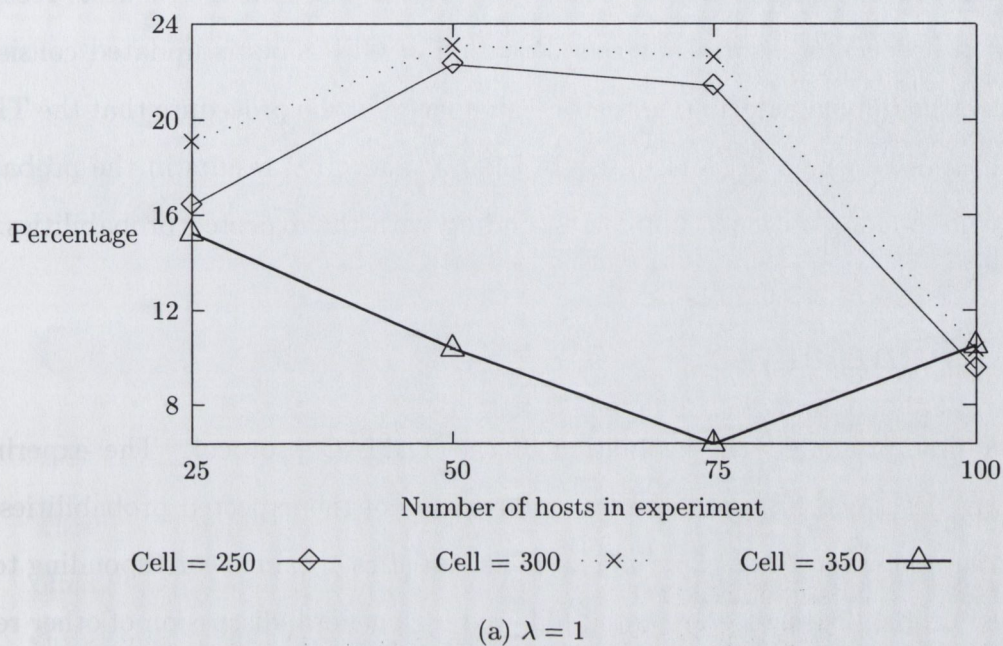


Fig. 5.13: Percentage of failed slots detected as having failed

number of atomic multicast messages outstanding (i.e., messages yet to be delivered). Therefore, enforcing some form of admission control based on the requests received in the CP is needed to ensure that the allocation of CFP slots is updated consistently. Secondly, the implementation, in the NS2 simulator, of the procedure that the TBMAC protocol uses to generate slots to use in the CP and CFP results in the probabilities of the time bounds obtained not corresponding with the expected probabilities.

5.5 Summary

This chapter presented the evaluation of the TBMAC protocol. The experimental setup was described first followed by a description of the expected probabilities based on the parameters to these experiments. The statistics obtained corresponding to these expected probabilities were presented followed by a general discussion of other relevant experimental results related to the TBMAC protocol.

Chapter 6

Conclusion

This thesis addressed the problem of providing time-bounded medium access to applications with guaranteed response time requirements in multi-hop ad hoc networks through the design of a new medium access control protocol called the Time Bounded Medium Access Control (TBMAC) protocol.

This chapter briefly describes the TBMAC protocol, summarises the main contribution of this thesis, places this work in perspective and suggests possible future work.

6.1 Summary

The TBMAC protocol uses a cellular structure to minimise the possible contention between mobile hosts in the ad hoc network and within each cell divides access to the wireless medium into two slotted time periods of known duration (called the CFP and CP). Mobile hosts in a cell to whom slots in the CFP have been allocated have predictable access to the wireless medium while mobile hosts that arrive in the cell must contend with each other in the CP to request the allocation of a CFP slot. This contention in the CP introduces the possibility of collisions between requesting mobile

hosts.

The TBMAC protocol uses an atomic multicast protocol to dynamically allocate CFP slots in a cell when a request for a CFP slot is received and to deallocate a CFP slot when a mobile host departs the cell. A mobile host in a cell updates the allocation of CFP slots by using the atomic multicast protocol to flood the allocation update to every other mobile host in the cell.

The TBMAC protocol incorporates a bootstrapping protocol to allow one or more mobile hosts to agree on a consistent initial allocation of CFP slots. At the start of the bootstrapping protocol, a mobile host generates a list of slots to transmit in and then communicates this list to other mobile hosts that are listening or also executing the bootstrapping protocol.

6.2 Contribution

As highlighted in the review of related work, previous MAC protocols for ad hoc networks have focused on maximising throughput and minimising average packet delay. This thesis described a new MAC protocol, called TBMAC, for use in multi-hop ad hoc networks whose goal is to provide, with known probability, time-bounded access to the wireless medium.

Associated with the allocation of a CFP slot are a set of time bounds that depend on whether the current view of the allocated CFP slots is consistent across the mobile hosts in the cell or not. These time bounds provided by the TBMAC protocol are not certain, therefore associated probabilities of achieving these time bounds were derived.

The probability of a mobile host not colliding with another mobile host when contending in the CP has been derived in chapter 3 of this thesis. This probability corresponds to the time bound in a non-empty cell being obtained given a particular probability of arrival into the cell. By increasing or decreasing the number of slots in

the CP, there is a corresponding increase or decrease in the expected probability of a mobile host choosing a distinct slot.

The bootstrapping protocol does not ensure that the mobile hosts agree on a consistent allocation of CFP slots with certainty within a known time bound. The bootstrapping protocol does, however, provide a known probability of this agreement occurring in the presence of an arbitrary number of mobile hosts executing this bootstrapping protocol and also in the presence of transient transmission errors. This second probability was also derived in chapter 3 of this thesis and it depends on the number of slots in the CFP, the size of the generated list used in the bootstrapping protocol and the number of rounds of the bootstrapping protocol. By modifying these parameters to the TBMAC protocol, this probability can be either increased or decreased as necessary.

It is important to note that the various parameters to the TBMAC protocol, such as the size of the CFP and CP and the number of rounds in the bootstrapping protocol, have an effect, firstly, on the probability of the time bounds in the TBMAC protocol being obtained and, secondly, on the duration of these time bounds. For example, increasing the size of the CP would result in a corresponding increase in the probability of a mobile host choosing a distinct slot to transmit a request in. However, this increase in size would also increase the duration of the time bound.

The evaluation of the TBMAC protocol highlighted a number of artefacts related to the implementation of the atomic multicast protocol in TBMAC. Firstly, the statistical properties of the algorithm to randomly choose either CP or CFP slots should be chosen to correspond to the expected probability. Secondly, the choice of cell size is an important factor to ensure that updates to the allocation of CFP slots can be executed consistently. Even with a judiciously chosen cell size, the need for admission control to ensure the correct functioning of the TBMAC protocol was highlighted by the evaluation. This requirement for admission control to ensure the consistent allocation of CFP slots highlights the conflict between the constraints of applications

with guaranteed response time requirements and the generality of multi-hop ad hoc networks.

6.3 Perspectives

The goal of the TBMAC protocol is to meet the requirements of these applications in an ad hoc network by imposing a small number of constraints on the mobile hosts. In limiting the number of constraints on the mobile hosts in the ad hoc network, the TBMAC protocol cannot provide time bounds to these mobile hosts with certainty. By imposing further restrictions on these mobile hosts it may be possible to provide these certain time bounds using TBMAC. For example, by restricting the speed and movement pattern of mobile hosts in the ad hoc network to eliminate the possibility of contention in the CP. However, imposing such a restriction is application specific and such a design decision can only be made by a layer above the TBMAC protocol.

In any case, the TBMAC protocol only provides a mechanism to gain access to the wireless medium in an ad hoc network. Higher layers above TBMAC would need to tailor the parameters of the TBMAC protocol to their particular needs and to build upon the functionality that TBMAC provides to enable applications with guaranteed response time requirements.

6.4 Future Work

Future work will investigate the possibility of providing time bounds with certainty on the execution of the bootstrapping protocol which is used to reach agreement on an initial allocation of slots that is consistent across the mobile hosts executing the bootstrapping protocol.

The ability to provide such a bootstrapping protocol with certain time bounds was

not immediately obvious at any stage during the course of the research that resulted in this thesis. This is in part due to the difficulty that two mobile hosts have in detecting that they are conflicting with each other. As highlighted in chapter 2, collisions occur at the receiver in multi-hop ad hoc networks thus giving no immediate feedback on the conflict to the transmitter until after the conflict has occurred. In addition, such a bootstrapping protocol must provide this time bound in the presence of the potentially random movement of mobile hosts in the ad hoc network.

An interesting exercise would be to investigate whether or not it is actually impossible to provide such certain time bounds in ad hoc networks. This mathematical proof would require a system model that reflects some of the problems posed by multi-hop ad hoc networks covered in chapter 2.

Bibliography

- [1] Ns-2 network simulator. <http://www.isi.edu/nsnam/ns/>.
- [2] Ns-allinone distribution version 2.1b9a. <http://www.isi.edu/nsnam/dist>.
- [3] OTcl - MIT Object Tcl. <ftp://ftp.tns.lcs.mit.edu/pub/otcl/README.html>.
- [4] 802.11g News. "Gartner report on the future growth of 802.11 network usage". <http://www.80211gnews.com/publications/page354-484719.asp>, 2003.
- [5] N. Abrahamson. "Development of the ALOHANET". *IEEE Transactions on Information Theory*, 31(6):119–123, March 1985.
- [6] Zigbee Alliance. "Zigbee Overview". http://www.zigbee.org/zigbee_new/resources/ZigBeeOverview4.pdf, 2003.
- [7] E. Arikan. "*Multi-access in packet-radio networks*". MIT, 1982.
- [8] O. Babaoglu, R. Davoli, L. Giachini, and M. Baker. "Relacs: A communication infrastructure for constructing reliable applications in large-scale distributed systems", 1995.
- [9] L. Bao and J. J. Garcia-Luna-Aceves. "A New Approach to Channel Access Scheduling for Ad Hoc Networks". In *7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, 2001.

- [10] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. "MACAW: a media access protocol for wireless LAN's,". *ACM SIGCOMM'94*, pages 212–225., 1994.
- [11] K. P. Birman. "*Building Secure and Reliable Network Applications*". Manning Publications Co., 1996. ISBN 0137195842.
- [12] K. P. Birman and T. Joseph. "Reliable Communication in the Presence of Failures". *ACM Transactions on Computer Systems*, 5(1):47–76, February 1987.
- [13] K. P. Birman, A. Schiper, and P. Stephenson. "Lightweight Causal and Atomic Group Communication". *ACM Transactions on Computing Systems*, 9(3):272–314, August 1991.
- [14] J. Brunekreef, J.-P. Katoen, R. Koymans, and S. Mauw. "Design and analysis of dynamic leader election protocols in broadcast networks". *Distributed Computing*, 9(4):157–171, March 1997.
- [15] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazo. "An Implicit Prioritized Access Protocol for Wireless Sensor Networks". In *IEEE Real-Time Systems Symposium*, Austin, Texas, December 2002.
- [16] J. Chang and N. Maxemchuk. "Reliable Broadcast Protocols". *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.
- [17] I. Chlamtac and A. Farago. "Making transmission schedules immune to topology changes in multi-hop packet radio networks". *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.
- [18] I. Cidon and M. Sidi. "Distributed Assignment Algorithms for Multihop Packet Radio Networks". *IEEE Transactions on Computers*, 38(10):739–746, October 1989.

- [19] R. Coker. *“Rollercoasters: A Thrill-Seekers Guide to the Ultimate Scream Machines”*. Prentice Hall, 2000.
- [20] F. Cristian. “Synchronous Atomic Broadcast for Redundant Broadcast Channels”. In *“Journal of Real-time Systems”*, pages 195–212. Kluwer Academic Publishers, 1990.
- [21] F. Cristian. “Understanding fault-tolerant distributed systems”. *Communications of the ACM*, 34(2):56–78, 1991.
- [22] F. Cristian, H. Aghali, R. Strong, and D. Dolev. “Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement”. In *Proc. 15th Int. Symp. on Fault-Tolerant Computing (FTCS-15)*, pages 200–206, Ann Arbor, MI, USA, 1985. IEEE Computer Society Press.
- [23] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. “Epidemic Algorithms for Replicated Database Maintenance”. In *The Sixth Symposium on Principles of Distributed Computing*, pages 1–12, August 1987.
- [24] ETSI. European telecommunications standards institute. <http://www.etsi.org>, 2003.
- [25] T. Facchinetti, G. Buttazo, M. Caccamo, and L. Almeida. “Wireless real-time communication protocol for cooperating mobile units”. In *2nd International Workshop on Real-Time LANs in the Internet Age*, pages 81–84, Porto, Portugal, July 2002.
- [26] K. Fall. “ns Notes and Documentation”. *The VINT Project*, 2000.
- [27] C. Fetzer and F. Cristian. “Integrating external and internal clock synchronization”. *Real-Time Systems*, 12(2):123–171, 1997.

- [28] C. Fetzer and F. Cristian. "A highly available local leader election service". *IEEE Transactions on Software Engineering*, 25(5):603–618, September 1999.
- [29] J. J. Garcia-Luna-Aceves. "Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks,". In *ACM SIGCOMM'95*, pages 262–273., 1995.
- [30] D. Gifford. "Weighted Voting for Replicated Data". In *Seventh ACM Symposium on Operating Systems Principles*, pages 150–162, 1979.
- [31] J. Gray. "Notes on Database Operating Systems. Operating Systems: An Advanced Course". *Springer Verlag Lecture Notes in Computer Science*, 60:393–481, 1978.
- [32] Bluetooth Special Interest Group. "Bluetooth Core Specification 1.1". <http://www.bluetooth.org>, 2003.
- [33] G Grunsteidl and H. Kopetz. "A Reliable Multicast for Distributed Real-Time Systems". In *8th IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, USA, May 1991. Technical University Vienna.
- [34] I. Gupta, R. van Renesse, and K. P. Birman. "A Probabilistically Correct Leader Election Protocol for Large Groups". In *International Symposium on Distributed Computing*, pages 89–103, 2000.
- [35] IEEE. "IEEE std 802.11 - wireless lan medium access control (MAC) and physical layer (PHY) specifications", 1997.
- [36] O. Klein J. Peetz, A. Hettich. "HiperLAN/2 Ad Hoc Network Configuration By CC Selection". In *Proc. of the European Wireless 2000 (EW2000)*, pages 1–6, Dresden, Germany, September 2000.
- [37] M Johnsson. "HIPERLAN/2 - The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band". <http://www.hiperlan2.com>, 2003.

- [38] J.H. Ju and V.O.K. Li. "An optimal topology-transparent scheduling method in multihop packet radio networks". *IEEE/ACM Transactions on Networking*, 6(3):298–306, June 1998.
- [39] M. F. Kaashoek and A. S. Tanenbaum. "Group Communication in the Amoeba Distributed Operating System". In *Eleventh International Conference on Distributed Computing Systems*, pages 222–230, New York, 1991. IEEE Computer Society Press.
- [40] J. Kaiser and M. A. Livani. "Achieving Fault-Tolerant Ordered Broadcasts in CAN". In *European Dependable Computing Conference*, pages 351–363, 1999.
- [41] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. "Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints". In *International Conference on Mobile Computing and Networking (MOBICOM)*, August 2001.
- [42] P. Karn. "MACA - a new channel access method for packet radio.". In *ARRL/CRRL Amateur Radio 9th Computer Conference*, pages 134–140, Ontario, Canada, 1990.
- [43] M. Kennedy. *"The Global Positioning System and GIS : an introduction"*. Ann Arbor Press, 1996.
- [44] D. E. Knuth. *Seminumerical Algorithms*. Addison-Wesley, 1981.
- [45] A Konstantinos. "Space-Time Division Multiple Access (STDMA) and Coordinated, Power-Aware MACA for Mobile Ad Hoc Networks". In *IEEE Symposium on Ad Hoc Wireless Networks (SAWN2001)*, San Antonio, Texas, 2001.
- [46] H. Kopetz. "The Time Triggered Model of Computation". In *19th Real-Time Systems Symposium*. Technical University Vienna, 1998.

- [47] H. Kopetz, M. Braun, C. Ebner, A. Krueger, D. Millinger, R Nossal, and A. Schedl. "The Design of Large Real-Time Systems: The Time Triggered Approach". In *16th Real-Time Systems Symposium*. Technical University Vienna, 1995.
- [48] H. Kopetz and W. Ochsenreiter. "Clock Synchronisation in Distributed Real-Time Systems". *IEEE Transactions on Computers*, 36(8):933–940, December 1987.
- [49] M. A. Livani, J. Kaiser, and W. Jia. "Scheduling Hard and Soft Real-Time Communication in Controller Area Network (CAN). In *23rd IFAC/IFIP Workshop on Real Time Programming*, Shantou, China, June 1998.
- [50] H. Luo, S. Lu, and V. Bharghavan. "A new model for packet scheduling in multihop wireless networks". In *International Conference on Mobile Computing and Networking (MOBICOM)*, August 2000.
- [51] D. Malki, Y. Amir, D. Dolev, and S. Kramer. "The Transis Approach to High Availability Cluster Communication". Technical Report CS94-14, 1994.
- [52] M. J. Markowski and A. S. Sethi. "Fully Distributed Wireless MAC Transmission of Real-Time Data". In *Fourth IEEE Real-Time Technology and Applications Symposium*, June 1998.
- [53] P. M. Melliar-Smith, L. E. Moser, and V. Agrawala. "broadcast protocol for distributed systems.". *IEEE Transactions on Parallel Distributed Systems.*, 1:17–25, 1990.
- [54] B. A. Miller and C. Bisdikian. *"Bluetooth Revealed"*. Prentice Hall, 2001.
- [55] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. "Achieving MAC layer fairness in wireless packet networks". In *International Conference on Mobile Computing and Networking (MOBICOM)*, August 2000.

- [56] J. Ousterhout. "Scripting: Higher-level programming for the 21st century". *IEEE Computer*, (3):23–30, 1998.
- [57] T. Ozugur, M. Naghshineh, P. Kermani, C. M. Olsen, B. Rezvani, and J. A. Copeland. "Balanced media access methods for wireless networks". In *International Conference on Mobile Computing and Networking (MOBICOM)*, October 1998.
- [58] M. Peraldi and J. Decotignie. "Combining real-time features of local area networks FIP and CAN". In *Proceedings of the 2nd International CAN Conference*, pages 8.11–8.21, London, England, 1995.
- [59] C. Petrioli, S. Basagni, and I. Chlamtac. "Configuring BlueStars: Multi-hop Scatternet Formation for Bluetooth Networks". *IEEE Transactions on Computers*, 52(6), 2003.
- [60] S. Poledna. "Fault tolerance in safety critical automotive applications: Cost of agreement as a limiting factor". In *25th International Symposium on Fault-Tolerant Computing Systems*, pages 73–82, Pasadena, California, 1995.
- [61] M. Purser. *"Introduction to error-correcting codes"*. Artech House, 1993.
- [62] M. Purser. *"Secure data networking"*. Artech House, 1995.
- [63] M. Rahnema. "Overview of the GSM System and Protocol Architecture". *IEEE Communications Magazine*, 31(4):92–100, April 1993.
- [64] T. S. Rappaport. *"Wireless Communications: Principle and Practice"*. Prentice Hall, 1996.
- [65] Robert Bosch GmbH. "CAN specification version 2.0", 1991.

- [66] L Roberts. "Extensions of Packet Communication Technology to a Hand Held Personal Terminal". In *Spring Joint Computer Conference AFIPS*, pages 295–298, 1972.
- [67] J. Rufino and P. Verissimo. "Enforcing Real-Time Behaviour on LAN-based Protocols". In *10th IFAC Workshop on Distributed Control Systems*, Semmering, Austria, September 1991.
- [68] J. Rufino and P. Verissimo. "A Study on the Inaccessibility Characteristics of ISO 8802-5 Token-Ring LAN,". Technical report, INESC, Lisboa, Portugal, 1992.
- [69] J. Rufino, P. Verissimo, G. Arroz, C. Almeida, and L. Rodrigues. "Fault Tolerant Broadcasts in CAN". In *28th IEEE International Symposium on Fault-Tolerant Computing*, Munich, Germany, 1998.
- [70] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. Lemaire. "Distributed topology construction of Bluetooth personal area networks". In *Proceedings of INFOCOM*, 2001.
- [71] J. F. Shoch and J. F. Hupp. "Measured Performance of an Ethernet Local Network". *Communications of the ACM*, 23(12):711–721, December 1980.
- [72] G. Singh. "Real-time Leader Election". *Information Processing Letters*, 49:57–61, 1994.
- [73] G. Singh. "Leader Election in the Presence of Link Failures". In *IEEE Transactions on Parallel and Distributed Computing*, volume 7, March 1996.
- [74] J. L. Sobrinho and A. S. Krishnakumar. "Real-time Traffic over the IEEE 802.11 Medium Access Control Layer". Technical report, Bell Labs, 1996.
- [75] W. Stallings. *"Data and Computer Communications"*. Prentice Hall, 2000.

- [76] W. Stallings. *"Local and Metropolitan Area Networks"*. Prentice Hall, 2000.
- [77] B. Stroustrup. *"The C++ Programming Language"*. Addison-Wesley, Reading, Massachusetts, USA, 1991.
- [78] H. A. Taha. *"Operations Research: An Introduction"*. Prentice Hall, 1997.
- [79] H. Takagi and L. Kleinrock. "Throughput Analysis of Persistent CSMA Systems". *IEEE Transactions on Communications*, 33(7), July 1985.
- [80] A. S. Tanenbaum. *"Computer Networks"*. Prentice Hall, 1996.
- [81] F. Tobagi and L. Kleinrock. "Packet Switching in Radio Channels: Part II-The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy-Tone Solution". *IEEE Transactions on Communications*, 23(12):1417-1433, December 1975.
- [82] J. Bao-yen Tsui. *"Fundamentals of global positioning system receivers : a software approach"*. Wiley, 2000.
- [83] P. Verissimo, L. Rodrigues, and A. Casimiro. "Cesiumspray: a precise and accurate global time service for large-scale systems", May 1997.
- [84] M. A. Visser and M. El Zarki. "Voice and Data Transmission over an 802.11 Wireless Network". In *PIMRC'95*, pages 648-652, September 1995.
- [85] W Zhao, J. A. Stankovic, and K. Ramamritham. "A window protocol for the transmission of time-constrained messages". *IEEE Transactions on Computers*, 39(9):1186-1203, September 1990.